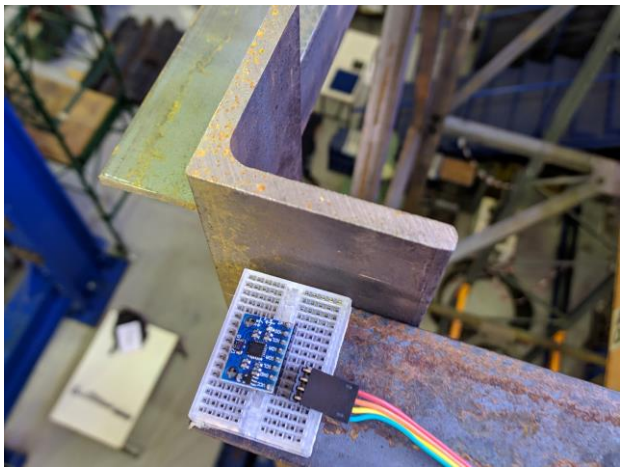
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Πολιτικών Μηχανικών
Εργαστήριο Μεταλλικών Κατασκευών

Προγραμματισμός επιταχυνσιομέτρου βασισμένου στη πλατφόρμα Arduino για τον υπολογισμό των ιδιομορφικών χαρακτηριστικών απλών μεταλλικών φορέων



ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βασίλειος Γ. Θωμάς

Επιβλέπων: Βαμβάτσικος Δημήτριος

Αθήνα, Σεπτέμβριος 2019

ΕΜΚ ΔΕ 2019/19

Θωμάς Β. Γ. (2019).
Προγραμματισμός επιταχυνσιομέτρου βασισμένου στη πλατφόρμα Arduino για τον υπολογισμό των ιδιομορφικών χαρακτηριστικών απλών μεταλλικών φορέων
Διπλωματική Εργασία ΕΜΚ ΔΕ 2019/19
Εργαστήριο Μεταλλικών Κατασκευών, Εθνικό Μετσόβιο Πολυτεχνείο, Αθήνα.

Thomas V. G. (2019).
Arduino platform based accelerometer programming for the calculation of the modal characteristics of simple metal structures
Diploma Thesis EMK ΔΕ 2019/19
Institute of Steel Structures, National Technical University of Athens, Greece

Πίνακας περιεχομένων

Περίληψη 2

Abstract 3

Ευχαριστίες 4

1 Εισαγωγή 5
1.1 Γενικά 5
1.2 Σκοπός 6
1.3 Οργάνωση περιεχομένων 7

2 Παρουσίαση εξαρτημάτων και λογισμικού 8
2.1 Επιταχυνσιόμετρο 8
2.2 Σειριακή επικοινωνία και δίαυλος I2C 9
2.3 Η πλατφόρμα Arduino 9
2.4 Η πλατφόρμα Teensy 12
2.5 Επεξεργασία δεδομένων σε Python 14
2.5.1 Ανάπτυξη αλγορίθμου σε γλώσσα Python 14
2.5.2 Εύρεση συχνοτήτων ταλάντωσης μέσα από επιταχυνσιογραφήματα 18
2.5.3 Υπολογισμός συντελεστή απόσβεσης μέσα από επιταχυνσιογραφήματα 20
2.5.4 Παραλλαγές αλγορίθμου 23

3 Αναγνώριση κυρίων ιδιοσυχνοτήτων ταλάντωσης και συντελεστών απόσβεσης μεταλλικού ελάσματος 25
3.1 Εισαγωγή 25
3.2 Περιγραφή φορέα 25
3.3 Δεδομένα μετρήσεων και αποτελέσματα 27
3.4 Συμπεράσματα 31

4 Αναγνώριση κύριας ιδιοσυχνότητας και συντελεστή απόσβεσης ταλάντωσης μεταλλικού προβόλου 32
4.1 Εισαγωγή 32
4.2 Από τη θεωρία στη πράξη 32
4.3 Πειραματικός υπολογισμός δυσκαμψίας της δοκού περί τον ισχυρό άξονα 34
4.4 Πρώτη σειρά μετρήσεων 38
4.5 Δεύτερη σειρά μετρήσεων 39
4.6 Τρίτη σειρά μετρήσεων 40
4.7 Εύρεση συντελεστή απόσβεσης ταλάντωσης 41
4.8 Εύρεση κύριας συχνότητας ταλάντωσης υπό μικρότερες δυνάμεις προεντάσεως κοχλία στην στήριξη 42
4.9 Συμπεράσματα 45

5 Συμπεράσματα 48

6 Βιβλιογραφία 49

Παράρτημα Α. Κώδικας λειτουργίας αισθητήρα 50

Παράρτημα Β. Κώδικας επικοινωνίας μεταξύ αισθητήρα και Python 52

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΕΤΑΛΛΙΚΩΝ ΚΑΤΑΣΚΕΥΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
ΕΜΚ ΔΕ 2019/19

Προγραμματισμός επιταχυνσιομέτρου βασισμένου στη πλατφόρμα Arduino για τον υπολογισμό των ιδιομορφικών χαρακτηριστικών απλών μεταλλικών φορέων

Θωμάς Β. Γ. (Επιβλέπων: Βαμβάτσιμος Δ.)

Περίληψη

Στα πλαίσια της παρακολούθησης κατασκευών με τη χρήση αισθητήρων έγινε μια προσπάθεια για την ανάπτυξη ενός ολοκληρωμένου συστήματος που θα καταγράφει και ταυτόχρονα θα επεξεργάζεται ταλαντώσεις που προκαλούνται από διάφορα αίτια. Από την χρονοϊστορία των αναπτυσσόμενων επιταχύνσεων των ταλαντώσεων αυτών μπορεί να υπολογιστεί η συχνότητα με την οποία εκτελούνται καθώς και ο συντελεστής με τον οποίο αποσβένονται. Τέτοιου είδους συστήματα υπάρχουν στην αγορά αλλά πολλές φορές η τιμή τους αποτρέπει την εφαρμογή τους. Για αυτόν το λόγο, αρχικός στόχος ήταν να παραμείνει χαμηλό το κόστος των εξαρτημάτων που θα αποτελούσαν το σύστημα αυτό. Επιλέχτηκε ένα επιταχυνσιόμετρο και δοκιμάστηκαν δύο μικροεπεξεργαστές, Arduino και Teensy, για την τροφοδοσία του αισθητήρα και την συλλογή των μετρήσεων. Για την επεξεργασία των δεδομένων γράφτηκε κώδικας σε γλώσσα προγραμματισμού Python και προέκυψαν τρεις εκδοχές του. Κατά τον προγραμματισμό λήφθηκαν υπόψη δύο παράμετροι, η αυτοματοποίηση της υπολογιστικής διαδικασίας και η ανάγκη για επίτευξη υψηλών συχνοτήτων καταγραφής των μετρήσεων.

Σε πρώτη φάση το σύστημα ελέγχθηκε σε ένα μεταλλικό έλασμα που στηριζόταν ως πρόβολος. Δοκιμάστηκε η απόκριση του βασικού αλγορίθμου σε μια σειρά διαφορετικών δυσκαμψιών και τα αποτελέσματα συγκρίθηκαν με τα αντίστοιχα θεωρητικά. Ακολούθησαν δοκιμές σε ένα ρεαλιστικότερο μοντέλο. Σε μια μονοπροέχουσα δοκό συγκρίθηκαν οι τιμές που εξάγονται από τις τρείς εκδοχές του κώδικα και προέκυψε η ιδανικότερη λύση. Τέλος, αξιολογήθηκε η ικανότητα του αλγορίθμου να εντοπίζει πιθανές βλάβες. Προκαλώντας αλλαγές στη συνθήκη στήριξης της δοκού και έχοντας εντοπίσει τις αλλαγές αυτές πειραματικά, έγινε μια σειρά από επιπλέον καταγραφές.

Συμπερασματικά, παρά το χαμηλό κόστος του αισθητήρα και των μικροεπεξεργαστών, οι συχνότητες ταλάντωσης εντοπίστηκαν με αρκετή ακρίβεια και οι τιμές των συντελεστών απόσβεσης είχαν μικρή απόκλιση μεταξύ τους. Σε ότι αφορά όμως την ανίχνευση βλαβών τα αποτελέσματα δεν ήταν ικανοποιητικά καθώς δεν φάνηκε να επηρεάζονται οι καταγραφές από τις επιβαλλόμενες αλλαγές.

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
FACULTY OF CIVIL ENGINEERING
INSTITUTE OF STEEL STRUCTURES

DIPLOMA THESIS
ΕΜΚ ΔΕ 2019/19

Arduino platform based accelerometer programming for the calculation of the modal characteristics of simple metal structures

Thomas V. G. (supervised by Vamvatsikos D.)

Abstract

For sure in structural health monitoring through sensors, an attempt was made to develop an integrated system that can record and simultaneously process vibrations caused by various sources. The frequencies as well as the damping rate of these vibrations can be calculated from the time history of the observed accelerations. Such systems exist in the market but often their price prevents them from being implemented. For this reason, the original aim was to keep the cost of the components of this system low. An accelerometer was selected and two microprocessors, Arduino and Teensy, were tested to power the sensor and collect the measurements. The data was processed via software implemented in the Python programming language and three versions emerged. Two parameters were taken into account in the software development, the automation of the computational process and the need to achieve high sampling frequencies.

Initially the system was tested on a metal plate mounted as a cantilever. The response of the basic algorithm to a series of different stiffnesses was tested and the results were compared with the corresponding theories. Tests on a more realistic model followed. In a one-way beam the values extracted from the three versions of the code were compared and the most ideal solution was found. Finally, the ability of the algorithm to detect possible failures was evaluated. Causing changes to the beam support condition and having identified these changes experimentally, a number of additional recordings were made.

In conclusion, despite the low cost of the sensor and microprocessors, the vibration frequencies were accurately detected while the damping rate estimates came with slightly higher errors. However, in the fault detection tests the results were not satisfactory as the recordings did not appear to be affected by the changes made.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα της εργασίας, κ. Δημήτριο Βαμβάτσικο, Επίκουρο Καθηγητή της Σχολής Πολιτικών Μηχανικών ΕΜΠ, για την βοήθεια που μου προσέφερε όλο αυτό το διάστημα. Είναι ιδιαίτερη τιμή για εμένα που μου εμπιστεύτηκε την ανάθεση του συγκεκριμένου θέματος. Επίσης, θα ήθελα να ευχαριστήσω τον Δρ. Ξενοφώντα Λιγνό για όλη την καθοδήγηση και τις γνώσεις που μου προσέφερε πάνω στο ηλεκτρολογικό και προγραμματιστικό κομμάτι της εργασίας. Επιπλέον, θα ήθελα να ευχαριστήσω και το υπόλοιπο προσωπικό του Εργαστηρίου Μεταλλικών Κατασκευών καθώς με δέχτηκαν στον χώρο εργασίας τους και με βοήθησαν σε οτιδήποτε χρειάστηκα. Ήταν μια πολύ ωραία εμπειρία για μένα το να βρίσκομαι στον χώρο του εργαστηρίου και να δω πράγματα που δεν θα είχα αλλιώς τη δυνατότητα.

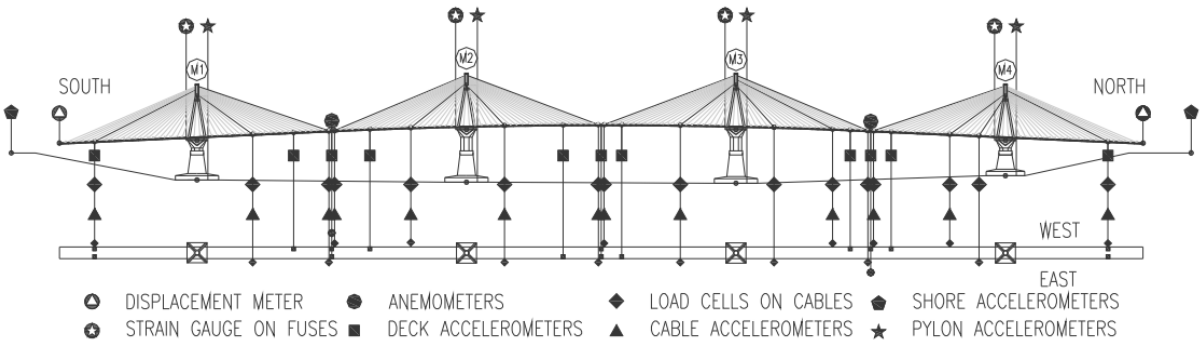
Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου που έχει κάνει παραπάνω από όσα χρειάζονται για μένα και με στηρίζει σε όλες μου τις αποφάσεις.

1 Εισαγωγή

1.1 Γενικά

Η ζωή όλων των τεχνικών έργων χαρακτηρίζεται από τρεις φάσεις, τη μελέτη, την κατασκευή και τη συντήρηση. Έχει παρατηρηθεί, όμως, πως από όλα αυτά η συντήρηση είναι κάτι που δεν εφαρμόζεται πολλές φορές. Ειδικότερα όταν πρόκειται για κατασκευές μεγάλης σπουδαιότητας, όπως γέφυρες και νοσοκομεία, οι επιπτώσεις της παραμέλησής τους μετά την κατασκευή τους μπορεί να είναι μοιραίες. Κατά καιρούς έχουν συμβεί ατυχήματα τα οποία είχαν ανυπολόγιστες συνέπειες για την κοινωνία και την οικονομία χωρών. Πέρα από αυτό, η σωστή συντήρηση ενός τεχνικού έργου μπορεί να μειώσει το κόστος μιας επερχόμενης βλάβης, αλλά και να συντελέσει στη μακροζωία του. Απαραίτητη προϋπόθεση για την σωστή συντήρηση του έργου είναι να υπάρχει συστηματική παρακολούθησή του. Η παρακολούθηση του έργου μπορεί να γίνεται επί τόπου με την χρήση ειδικών οργάνων, αλλά μπορεί να γίνεται και εξ αποστάσεως με την χρήση αισθητήρων (Εικόνα 1.1). Η χρήση επιταχυνσιομέτρων ενδείκνυται για την ανίχνευση οποιασδήποτε αλλαγής στα χαρακτηριστικά ενός υλικού ή στις συνοριακές συνθήκες των μελών μιας κατασκευής. Με τον τρόπο αυτό εξοικονομείται κόστος από την επί τόπου παρακολούθηση του έργου, ενώ ταυτόχρονα υπάρχει δυνατότητα για άμεσο εντοπισμό βλαβών. Δεν είναι τυχαίο πως μια από τις σπουδαιότερες γέφυρες στην Ελλάδα, η γέφυρα Ρίου-Αντιρρίου, διαθέτει συνολικά 372 αισθητήρες διαφορετικών τύπων ικανούς να ανιχνεύουν οποιαδήποτε εκδήλωση σεισμού, ισχυρών ανέμων ή άλλων φαινομένων (Εικόνα 1.2). Ένας «απλός» τρόπος αναγνώρισης βλαβών είναι συγκρίνοντας τα αποτελέσματα καταγραφών πριν και μετά από την εκδήλωση κάποιας διαταραχής. Για παράδειγμα, η εμφάνιση ενός σεισμού μπορεί να μειώσει τη δυσκαμψία της κατασκευής και κατά συνέπεια να αλλάξουν οι ιδιοσυχνότητες ταλάντωσής της. Ακόμα πιο προηγμένα συστήματα έχουν την δυνατότητα να αναγνωρίσουν που ακριβώς βρίσκονται αυτές οι βλάβες. Σύγκριση μπορεί να γίνεται και μεταξύ των χαρακτηριστικών του έργου όπως προκύπτουν από το υπολογιστικό μοντέλο πριν την κατασκευή με αυτά που προκύπτουν από καταγραφές των αισθητήρων. Δεδομένου ότι κατά την μελέτη υπάρχουν στοιχεία που δεν λαμβάνονται υπόψη όπως τοιχοπληρώσεις, έπιπλα, οχήματα, τα οποία υποκαθίστανται από μια σειρά συντελεστών ασφαλείας, υπάρχουν περιπτώσεις που μπορεί οι φυσικές ιδιότητες της κατασκευής να μην είναι αυτές που είχαν προβλεφθεί. Ως αποτέλεσμα αυτού, το υπολογιστικό μοντέλο της κατασκευής μπορεί να τροποποιείται ώστε να ανταποκρίνεται στη πραγματικότητα και να είναι διαθέσιμο σε περίπτωση συντήρησης, επισκευής ή και ενίσχυσης μετά από καιρό. Με άλλα λόγια, η χρήση αισθητήρων και ειδικότερα επιταχυνσιομέτρων καθίσταται αναγκαία για όλες τις υπάρχουσες κατασκευές μεγάλης σπουδαιότητας.

Εικόνα 1.1: Επιταχυνσιόμετρο σε καλωδιωτή γέφυρα (από http://sixense-systems.com/eversense/)



Εικόνα 1.2: Αισθητήρες κατά μήκος της γέφυρας του Ρίου-Αντιρρίου για την βέλτιστη ανίχνευση ισχυρών ανέμων και σεισμών (από Papanikolas P., Stathopoulos-Vlamis A., Panagis A.)

1.2 Σκοπός

Δεδομένου ότι η εφαρμογή των αισθητήρων σε έργα πολιτικού μηχανικού έχει μεγάλη σπουδαιότητα, γιατί είναι λίγες οι κατασκευές που γίνεται χρήση τους; Ο κυριότερος λόγος είναι το κόστος. Κόστος όχι μόνο για την αγορά των αισθητήρων και ενός συστήματος διαχείρισής τους, αλλά και για τη συντήρηση τους. Το πλήθος των αισθητήρων αυτών μπορεί να ποικίλει ανάλογα με το μέγεθος και τη σπουδαιότητα της κατασκευής. Σε μια προσπάθεια μείωσης αυτού του κόστους, φαίνεται αισιόδοξη η χρήση μικροεπεξεργαστών σε συνδυασμό με αισθητήρες χαμηλού κόστους που κυκλοφορούν στην αγορά. Στη παρούσα εργασία γίνεται διερεύνηση της αξιοπιστίας ενός συστήματος μικροεπεξεργαστή και επιταχυνσιομέτρου για τον υπολογισμό της κύριας συχνότητας ταλάντωσης διαφόρων μοντέλων. Στόχος ήταν η ανάπτυξη ενός αυτοματοποιημένου αλγορίθμου εισαγωγής και εξαγωγής δεδομένων. Ως εισαγόμενα δεδομένα θεωρούνται οι μετρήσεις του αισθητήρα, ενώ ως εξαγόμενα η συχνότητα ταλάντωσης καθώς και ο

συντελεστής απόσβεσης στην εξεταζόμενη διεύθυνση. Σε μια περαιτέρω διερεύνηση γίνεται μια προσπάθεια για ανίχνευση βλαβών μέσα από πιθανές αλλαγές στη συχνότητα ταλάντωσης του φορέα.

1.3 Οργάνωση περιεχομένων

Στο δεύτερο κεφάλαιο γίνεται μια παρουσίαση των εργαλείων που χρησιμοποιήθηκαν για την παρούσα εργασία. Συγκεκριμένα, οι μικροεπεξεργαστές που δοκιμάστηκαν καθώς και το επιταχυνσιόμετρο. Στη συνέχεια, γίνεται μια ανασκόπηση της φιλοσοφίας γύρω από την οποία αναπτύχθηκε ο κώδικας για την επικοινωνία μεταξύ αισθητήρα, μικροεπεξεργαστή και υπολογιστή καθώς και του τρόπου με τον οποίο παρουσιάζονται τα οποιαδήποτε αποτελέσματα.

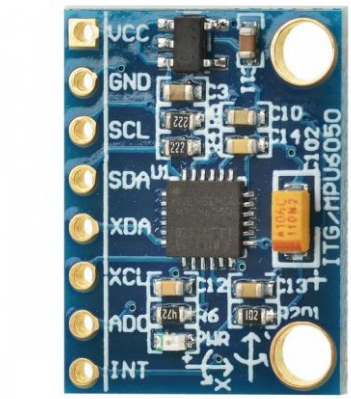
Στο επόμενο κεφάλαιο, γίνεται η πρώτη δοκιμή του συστήματος που αναπτύχθηκε. Για τον σκοπό αυτό επιλέχτηκε ένας μεταλλικός χάρακας, ο οποίος στερεώθηκε σε ένα έδρανο στο ένα άκρο του. Η δυνατότητα για τροποποίηση του ελεύθερου μήκους του συγκεκριμένου μοντέλου έδωσε τη δυνατότητα να ελεγχθεί ο αισθητήρας για διαφορετικές συχνότητες ταλάντωσης. Οι συχνότητες αυτές, αρχικά, υπολογίστηκαν μέσα από θεωρητικές σχέσεις και αποτέλεσαν μέτρο σύγκρισης για τα αποτελέσματα του αλγορίθμου. Επιπλέον, δοκιμάστηκε πλήθος συχνοτήτων καταγραφής των τιμών των επιταχύνσεων ώστε να εξεταστεί πως επιδρούν στις εκτιμώμενες συχνότητες ταλάντωσης.

Στο τέταρτο και τελευταίο κεφάλαιο εξετάζεται ένας φορέας πιο κοντά στη πραγματικότητα των κατασκευών του πολιτικού μηχανικού. Επιλέχτηκε μια χαλύβδινη μονοπροέχουσα δοκός, διατομής IPE. Η δυσκολία για υπολογισμό της συχνότητας ταλάντωσής της περί τον ισχυρό της άξονα μέσα από κλειστούς τύπους, οδήγησε στην εκτέλεση μιας σειράς πειραμάτων για τον υπολογισμό της πραγματικής της δυσκαμψίας. Με γνώμονα τα πειραματικά αποτελέσματα μπόρεσαν και αξιολογήθηκαν τα εξαγόμενα δεδομένα του κώδικα. Τα σφάλματα που προέκυψαν οδήγησαν στην εξέταση άλλων δύο αλγορίθμων οι οποίοι αναπτύχθηκαν με στόχο την επίτευξη μιας αυξημένης και πιο σταθερής συχνότητας καταγραφής σε σχέση με τον αρχικό αλγόριθμο. Τέλος, δοκιμάστηκε η ικανότητα του αλγορίθμου στην ανίχνευση βλαβών. Μέσα από μια αλλαγή στη συνθήκη στήριξης του φορέα και κατά συνέπεια στη συχνότητα ταλάντωσής του, ο κώδικας κλήθηκε να εντοπίσει την αλλαγή αυτή.

2 Παρουσίαση εξαρτημάτων και λογισμικού

2.1 Επιταχυνσιόμετρο

Τα επιταχυνσιόμετρα είναι από τους πιο διαδεδομένους αισθητήρες που χρησιμοποιούνται για την παρακολούθηση της δυναμικής απόκρισης μιας κατασκευής. Με κριτήριο κυρίως την οικονομικότητα των εξαρτημάτων που θα χρησιμοποιηθούν στη παρούσα εργασία, αλλά και την ακρίβεια των αποτελεσμάτων τους, επιλέχτηκε ο αισθητήρας MPU 6050 (Εικόνα 2.1). Ο MPU 6050 είναι ένας ψηφιακός αισθητήρας και διαθέτει επιταχυνσιόμετρο και γυροσκόπιο τριών αξόνων αντιστοίχως, ενώ δίνει και τη δυνατότητα παρακολούθησης της θερμοκρασίας του περιβάλλοντος χώρου. Όλα τα αποτελέσματα που θα παρουσιαστούν παρακάτω προκύπτουν από τις μετρήσεις του επιταχυνσιομέτρου επομένως δεν θα γίνει κάποια περαιτέρω αναφορά στο γυροσκόπιο και τον μετρητή θερμοκρασίας. Σε ότι αφορά το επιταχύνσεις, ο χρήστης έχει τη δυνατότητα να ρυθμίσει το εύρος στο οποίο θέλει να λειτουργεί ο αισθητήρας μέσα από τέσσερις επιλογές. Ο αισθητήρας μπορεί να λειτουργεί στα ±2g, ±4g, ±8g, ή ±16g (Εικόνα 2.2). Για εφαρμογές πολιτικού μηχανικού το εύρος των ±2g είναι αρκετό. Επιπλέον, μικρότερο εύρος τιμών ισοδυναμεί με μεγαλύτερη ευαισθησία στις μετρήσεις και άρα ανίχνευση μεγαλύτερης γκάμας διεγέρσεων. Οι μετρήσεις καταγράφονται σε bytes και από εκεί μεταφέρονται μέσω του σειριακού πρωτοκόλλου επικοινωνίας I2C σε οποιαδήποτε πλατφόρμα μπορεί υποστηρίξει αυτόν τον τρόπο επικοινωνίας.



Εικόνα 2.1: Ο αισθητήρας MPU 6050 (από **https://www.cableworks.gr**)

Product Details

Part #	Gyro Full Scale Range	Gyro Sensitivity	Gyro Rate Noise	Accel Full Scale Range	Accel Sensitivity	Digital Output
UNITS:	(°/sec)	(LSB/°/sec)	dps/√Hz	(g)	LSB/g	
 MPU-6050	±250 ±500 ±1000 ±2000	131 65.5 32.8 16.4	0.005 0.005 0.005 0.005	±2 ±4 ±8 ±16	16384 8192 4096 2048	I²C

Εικόνα 2.2: Εύρη λειτουργίας και ευαισθησία επιταχυνσιόμετρου και γυροσκοπίου για τον αισθητήρα MPU 6050 όπως παρουσιάζονται από τον κατασκευαστή (από **https://www.invensense.com**)

2.2 Σειριακή επικοινωνία και δίαυλος I2C

Η σειριακή επικοινωνία είναι ένας τρόπος με τον οποίο τα δεδομένα μπορούν να αποστέλλονται από μια συσκευή σε μία άλλη. Η επικοινωνία γίνεται μέσω κάποιου διαύλου, ο οποίος αποστέλλει σε κάθε του βήμα ένα μόνο bit της πληροφορίας. Η τρόπος αυτός πλεονεκτεί της παράλληλης επικοινωνίας λόγω του μειωμένου πλήθους καλωδίων που απαιτούνται και της δυνατότητας διασύνδεσης συσκευών σε μεγάλες αποστάσεις χωρίς να χάνονται σημαντικά δεδομένα. Από τους πλέον διαδομένους διαύλους σειριακής επικοινωνίας είναι οι RS-232 (Recommended Standard), SPI (Serial Peripheral Interface) και I2C (Inter-Integrated Circuit). Το πρωτόκολλο RS-232 παρουσιάστηκε από την Electronic Industries Association και χρησιμοποιούταν για αρκετό καιρό έως και πρόσφατα στις σειριακές θύρες των ηλεκτρονικών υπολογιστών για τη σύνδεσή τους περιφερειακές συσκευές. Από την άλλη το πρωτόκολλο SPI αναπτύχθηκε από την Motorola και χρησιμοποιείται κυρίως σε επίπεδο πλακέτας ή για πολύ μικρές αποστάσεις.

Ο δίαυλος επικοινωνίας I2C, σε πρώτη φάση, αναπτύχθηκε από την Philips και σήμερα εντοπίζεται στις πλακέτες πολλών συσκευών που χρησιμοποιούνται στη καθημερινή ζωή. Για τη μεταφορά των δεδομένων απαιτούνται μόνο δύο γραμμές επικοινωνίας, μία γραμμή σειριακών δεδομένων (SDA) και μία γραμμή σειριακού ρολογιού (SCL) για τον συγχρονισμό των συσκευών που είναι συνδεδεμένες μεταξύ τους. Κάθε συσκευή είναι πομπός ή δέκτης ανάλογα με το αν δέχεται ή στέλνει δεδομένα. Επιπλέον, μπορούν να συνδεθούν μεταξύ τους περισσότεροι του ενός πομποί και δέκτες, ενώ η επικοινωνία τους μπορεί να είναι και αμφίδρομη. Για τις εφαρμογές που έγιναν χρησιμοποιήθηκε μόνο ένας αισθητήρας MPU 6050 και η μεταφορά των δεδομένων υλοποιούταν μέσω της σύνδεσής του με τις πλατφόρμες Arduino και Teensy, χωριστά σε κάθε περίπτωση.

2.3 Η πλατφόρμα Arduino

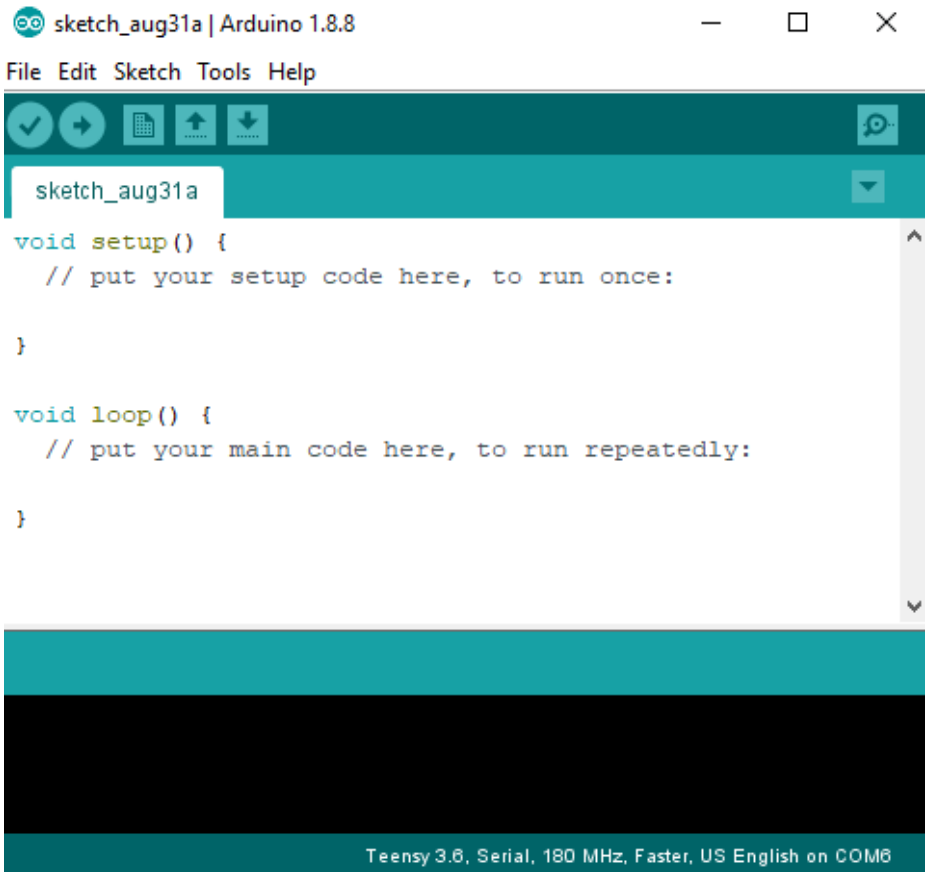
Ο Arduino είναι μια ηλεκτρονική πλατφόρμα (Εικόνα 2.3) που μπορεί μέσα από μια σχετικά εύκολη προγραμματιστική διαδικασία να διαβάζει δεδομένα (inputs) ή να εξάγει αποτελέσματα (outputs). Παρέχεται από την εταιρία που φέρει το ίδιο όνομα με την πλατφόρμα και αρχικά χρησιμοποιήθηκε για εκπαιδευτικούς σκοπούς στο Interaction Design Institute Ivrea της Ιταλίας. Μπορεί να καλύψει ένα μεγάλο εύρος εφαρμογών, από το άνοιγμα ενός απλού led μέχρι την ανάγνωση των μετρήσεων ενός επιταχυνσιομέτρου ή τη δημιουργία ενός συστήματος συναγερμού. Ο χρήστης μπορεί να προγραμματίσει τον Arduino μέσα από μια σειρά εντολών τις οποίες τις γράφει σε ένα προγραμματιστικό

περιβάλλον (Integrated Development Environment) που έχει αναπτυχθεί για τον σκοπό αυτό, το *Arduino Software (IDE),* σε γλώσσα C ή C++ (Εικόνα2.4). Στην παρούσα εργασία χρησιμοποιήθηκε η έκδοση 1.8.8.

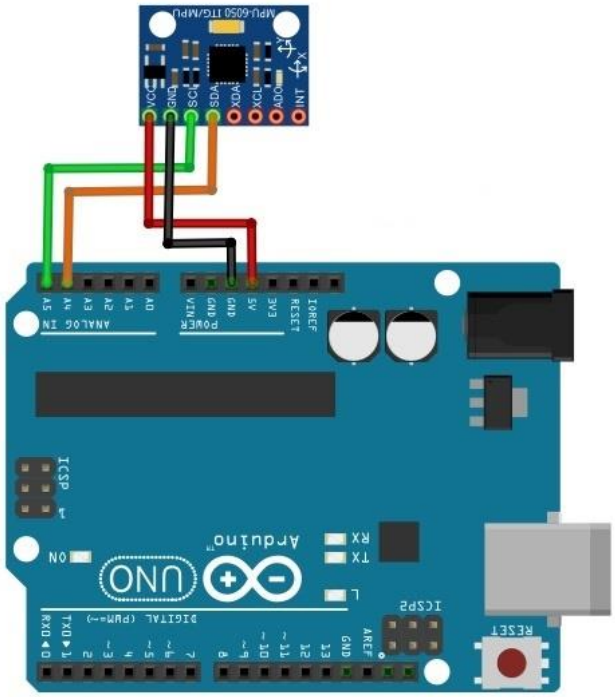
Η επιλογή της συγκεκριμένης πλατφόρμας έγινε για το χαμηλό της κόστος και την ευκολία προγραμματισμού της που παρέχει στον χρήστη. Πιο αναλυτικά, χρησιμοποιήθηκε το Aruduino UNO, η οποία κοστολογείται από την εταιρία στα 20 € (τιμή για το 2019). Βέβαια, απαραίτητη προϋπόθεση για να έχει νόημα η επιλογή του ήταν να υποστηρίζει το πρωτόκολλο I2C ώστε να μπορεί να γίνει η ανάγνωση των μετρήσεων του επιταχυνσιομέτρου. Από τη στιγμή που υποστηρίζεται ο επιθυμητός τρόπος επικοινωνίας μεταξύ αισθητήρα και πλατφόρμας μπορεί να γίνει η σύνδεση μεταξύ τους. Για την σύνδεσή τους χρειάζονται τέσσερα καλώδια τύπου αρσενικό σε αρσενικό, ένα για το ρεύμα (3,3 V), ένα για τη γείωση (ground) και τα δύο για την μεταφορά των δεδομένων (SDA και SDL) (Εικόνα 2.5). SCL (Serial Clock) είναι η γραμμή του ρολογιού που χρησιμοποιείται για τον συγχρονισμό των μεταφερόμενων δεδομένων μέσω του πρωτοκόλλου I2C και SDA (Serial Data) είναι η γραμμή για τη μεταφορά των δεδομένων.

Για τον προγραμματισμό της, θα πρέπει σε πρώτο στάδιο (set up) να οριστεί η διεύθυνση του αισθητήρα με την οποία θα καταλαβαίνει ο αλγόριθμος ότι γίνεται αναφορά σε αυτόν. Οι διευθύνσεις είναι μια σειρά από δεκαεξαδικούς αριθμούς με τους οποίους γνωστοποιείται στον μικροεπεξεργαστή η θέση της συσκευής με την οποία θα επικοινωνήσει, αλλά και τι δεδομένα θα ανταλλάξουν μεταξύ τους. Οι διευθύνσεις αυτές παρέχονται από τον κατασκευαστή του αισθητήρα και αποτελούν χαρακτηριστικό στοιχείο της επικοινωνίας με το πρωτόκολλο I2C. Επιπλέον, θα πρέπει να οριστεί το εύρος των τιμών που αναμένεται να μετρά το επιταχυνσιόμετρο. Στη συνέχεια, με τον ίδιο τρόπο εισάγονται και οι διευθύνσεις των εντολών που δίνουν τις τιμές των μετρήσεων μέσα σε μια επαναληπτική διαδικασία (loop) καθώς και ο τρόπος με τον οποίο θα παρουσιάζονται τα αποτελέσματα στον χρήστη (Εικόνα 2.5). Ο χρόνος μέσα στον οποίο θα πρέπει να πραγματοποιείται αυτή η επαναληπτική διαδικασία θα παίξει ιδιαίτερο ρόλο στη παρούσα εργασία καθώς σε επόμενα κεφάλαια θα γίνει αισθητό πως υπάρχει ένα ανώτερο όριο για το οποίο μπορεί να θεωρηθεί ότι οι μετρήσεις είναι αξιόπιστες.

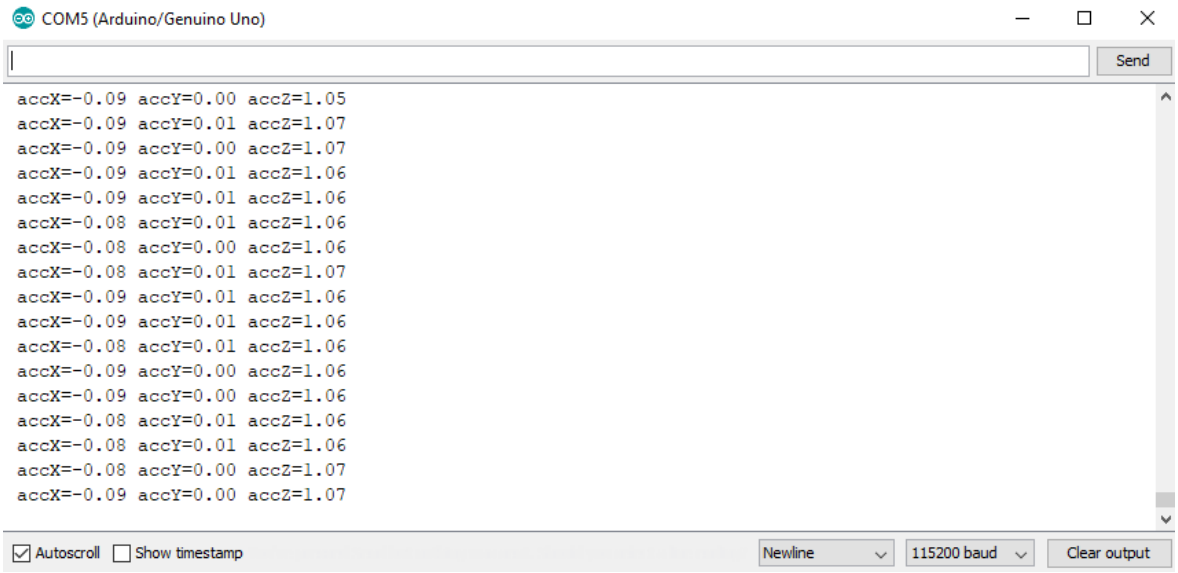
Εικόνα 2.3: Η πλατφόρμα Arduino Uno που χρησιμοποιήθηκε στη παρούσα εργασία



Εικόνα 2.4: Το περιβάλλον προγραμματισμού Arduino



Εικόνα 2.5: Σύνδεση του αισθητήρα MPU6050 με τον Arduino UNO (από theelectronics.co.in) στις θέσεις για το ρεύμα (3.3 V), τη γείωση (GND) και τη μεταφορά των δεδομένων (SDA και SCL).



Εικόνα 2.6: Παρουσίαση των μετρήσεων του επιταχυνσιομέτρου στους άξονες X, Y, Z, όπως αυτοί είναι ορισμένοι από τον κατασκευαστή. Τιμές συναρτήσει της επιτάχυνσης της βαρύτητας g.

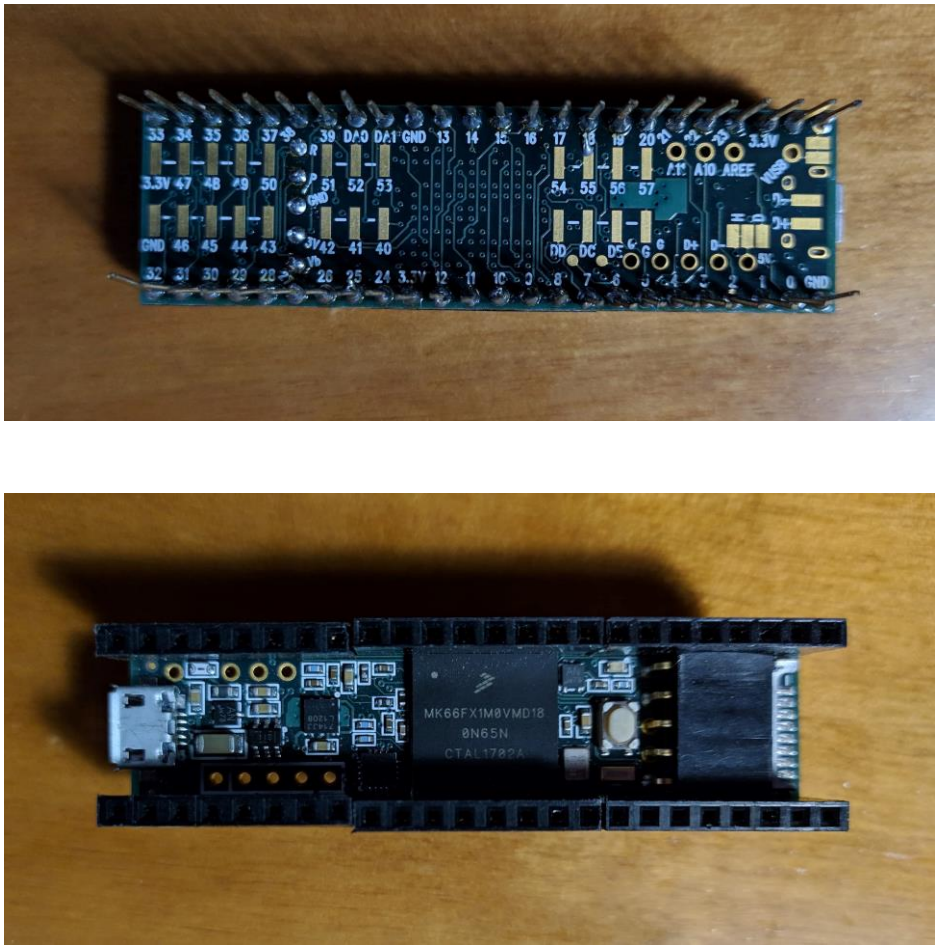
Τα αποτελέσματα των μετρήσεων μεταφέρονται στον υπολογιστή μέσω USB σειριακά και ο χρήστης μπορεί να ορίσει την ταχύτητα με την οποία θα πραγματοποιείται αυτή η επικοινωνία (baud rate). Στη προσπάθεια να αξιοποιηθούν οι μέγιστες δυνατότητες της πλατφόρμας, ορίστηκε η μέγιστη ταχύτητα, δηλαδή τα 115200 bits ανά δευτερόλεπτο. Παρόλο που μέχρι στιγμής ο Arduino φαίνεται να ικανοποιεί τον σκοπό της εργασίας, μειονεκτεί στο ότι ο επεξεργαστής του είναι 8-bit. Αυτό σημαίνει ότι κάθε χρονική στιγμή μπορεί και επεξεργάζεται 8 bits πληροφορίας. Αντιθέτως υπάρχουν άλλες πλατφόρμες που μπορούν να επεξεργαστούν τα δεδομένα με πολύ μεγαλύτερες ταχύτητες. Μια από αυτές είναι και ο Teensy.

2.4 Η πλατφόρμα Teensy

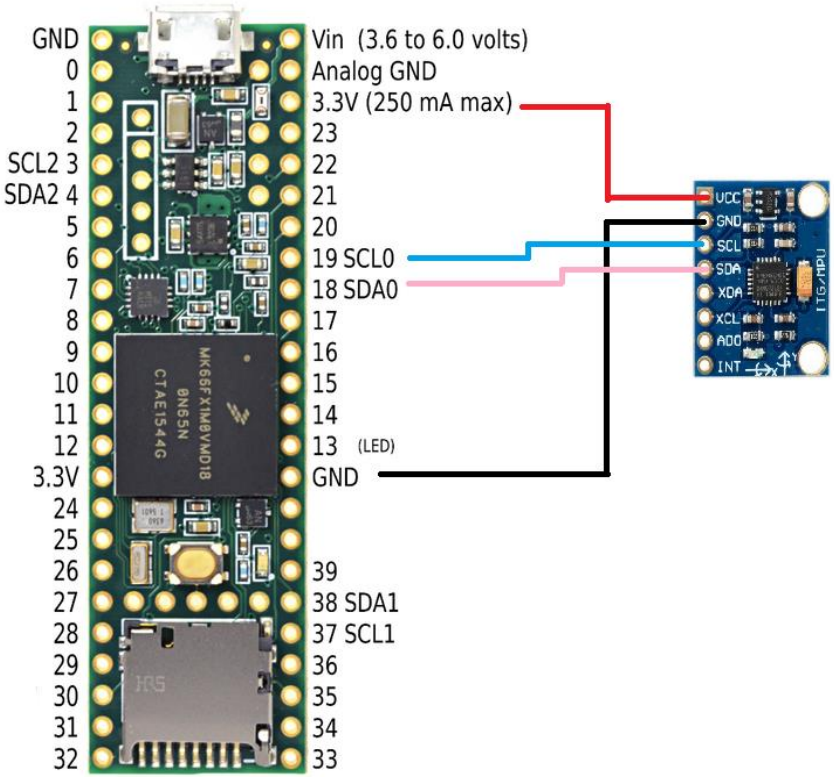
Ο Teensy, όπως και ο Arduino, είναι μια ηλεκτρονική πλατφόρμα που παρέχει τη δυνατότητα σε προγραμματιστές να αναπτύξουν πληθώρα εφαρμογών. Έχει ήδη αναφερθεί πως ένα από τα μεγαλύτερα πλεονεκτήματά του είναι οι γρήγορες ταχύτητες επεξεργασίας. Εκτός αυτού, έχει αρκετά μικρότερες διαστάσεις σε σχέση με άλλες πλατφόρμες, ενώ υπάρχει και η δυνατότητα αποθήκευσης δεδομένων σε κάρτα SD, μιας φορητής συσκευής διαδεδομένης για την αποθήκευση και τη μεταφορά ψηφιακών αρχείων. Σημαντικό για την παρούσα εργασία είναι και το γεγονός ότι μπορεί να υποστηρίξει 3 διαύλους επικοινωνίας I2C όπου στον κάθε ένα μπορούν να συνδεθούν από δύο συσκευές. Αυτό σημαίνει ότι μπορεί υποστηρίξει έως και έξι επιταχυνσιόμετρα αν και αυτό δεν βρίσκει εφαρμογή στη παρούσα εργασία. Ο Teensy μπορεί να προγραμματιστεί με το ίδιο λογισμικό που προγραμματίζεται και ο Arduino με αποτέλεσμα, στις περισσότερες των περιπτώσεων να μην χρειάζονται αλλαγές στον κώδικα για την επίτευξη της επικοινωνίας μεταξύ πλατφόρμας και αισθητήρα. Στη παρούσα εργασία χρησιμοποιήθηκε ο Teensy 3.6 (Εικόνα 2.7), ο οποίος διαθέτει έναν επεξεργαστή 32-bit,μπορεί δηλαδή να αποστείλει τετραπλάσιο μέγεθος πληροφορίας σε κάθε επανάληψη. Αν και από πλευράς τεχνικών χαρακτηριστικών ο Teensy πλεονεκτεί του Arduino, όπως

είναι αναμενόμενο, είναι μια πλατφόρμα συγκριτικά ακριβότερη. Συγκεκριμένα ο Teensy 3.6 κοστολογείται στη Ελλάδα στα 35 € (τιμή για το 2019).

Πριν τον προγραμματισμό της πλατφόρμας θα πρέπει να γίνει η σύνδεση της με τον αισθητήρα. Όπως και με τον Arduino, η λογική είναι ακριβώς η ίδια. Ένα καλώδιο για την παροχή ρεύματος τάσης 3,3 V, ένα για τη γείωση και άλλα δύο για την μεταφορά δεδομένων (Εικόνα 2.8). Στη συνέχεια πρέπει να εγκατασταθεί η προσθήκη Teensyduino στο πρόγραμμα Arduino, ώστε να μπορεί να γίνει ο προγραμματισμός του Teensy. Αφού ολοκληρωθεί και αυτό, η κωδικοποίηση της πλατφόρμας ακολουθεί την ίδια λογική με αυτή του Arduino, χωρίς να αλλάζει κάτι στις διάφορες εντολές.



Εικόνα 2.7: Ο Teensy 3.6 που χρησιμοποιήθηκε στη παρούσα εργασία (άνω και κάτω όψη)



Εικόνα 2.8: : Σύνδεση του αισθητήρα MPU6050 με τον Teensy 3.6 στις θέσεις για το ρεύμα (3.3 V), τη γείωση (GND) και τη μεταφορά των δεδομένων (SDA και SCL).

2.5 Επεξεργασία δεδομένων σε Python

2.5.1 Ανάπτυξη αλγορίθμου σε γλώσσα Python

Ο προγραμματισμός των Arduino και Teensy ήταν το βασικότερο στάδιο για τη δημιουργία ενός συστήματος που θα μπορεί να παίρνει τις καταγραφές ενός αισθητήρα και να τις παρουσιάζει στον χρήστη. Όμως, μέχρι στιγμής οι καταγραφές αυτές δεν μπορούν να αποθηκευτούν ή να επεξεργαστούν ώστε να δώσουν κάποια χρήσιμη πληροφορία για το αντικείμενο μελέτης. Για τον σκοπό αυτό, αναπτύχθηκε ένας αλγόριθμος σε γλώσσα Python και συγκεκριμένα στην έκδοση 3.7, ο οποίος όχι μόνο σώζει τις τιμές αυτές, αλλά και τις επεξεργάζεται, δημιουργώντας μια αυτοματοποιημένη υπολογιστική διαδικασία. Για τη συγγραφή του αλγορίθμου αυτού χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment) PyCharm, το οποίο παρέχεται δωρεάν από την εταιρία JetBrains.

Η παρουσίαση της λογικής του αλγορίθμου είναι απαραίτητη για την κατανόηση της μεθοδολογίας μέσω της οποίας προκύπτουν τα αποτελέσματα που παρουσιάζονται σε επόμενα κεφάλαια. Στόχος ήταν να μειωθεί όσο γινόταν η δουλεία του χρήστη κατά τη διάρκεια εκτέλεσης και επεξεργασίας μιας μέτρησης. Πριν την εκτέλεση οποιουδήποτε αλγορίθμου θα πρέπει ο αισθητήρας να έχει συνδεθεί με τον Arduino ή τον Teensy και να έχει ελεγχθεί αν υπάρχει επικοινωνίας μεταξύ τους. Αφού όλα είναι έτοιμα, ακολουθεί μια διαδικασία προετοιμασίας του αλγορίθμου για το συγκεκριμένο πείραμα. Για τον σκοπό αυτό αναπτύχθηκε ένας μικρότερος αλγόριθμος που υπολογίζει τη μέση τιμή των καταγραφών του επιταχυνσιομέτρου όταν ο φορέας βρίσκεται σε ισορροπία. Οι τιμές αυτές είναι απαραίτητες για να μπορεί να αξιολογεί και ο ίδιος ο αλγόριθμος αν εκτελείται ταλάντωση και αν ναι, πότε το σώμα διέρχεται από τη θέση ισορροπίας του. Ο χρήστης θα

πρέπει στη συνέχεια να τροποποιήσει τις τιμές των μεταβλητών του κύριου αλγορίθμου που αντιστοιχούν στις μέσες τιμές των επιταχύνσεων περί τους άξονες X, Y, Z με βάση τα αποτελέσματα που πήρε. Τώρα ο κύριος αλγόριθμος είναι έτοιμος να εκτελεστεί και να παρουσιάσει τα αποτελέσματα των μετρήσεων. Προφανώς, για να έχουν νόημα οι μετρήσεις θα πρέπει ο υπό εξέταση φορέας να διεγερθεί με οποιονδήποτε τρόπο ώστε να εκτελέσει μια μορφή ταλάντωσης τουλάχιστον σε έναν από τους τρεις άξονες μεταφοράς του επιταχυνσιομέτρου. Πατώντας το πλήκτρο εκτέλεσης του αλγορίθμου (run) ξεκινά η διαδικασία καταγραφής, αποθήκευσης και επεξεργασίας των τιμών του επιταχυνσιομέτρου.

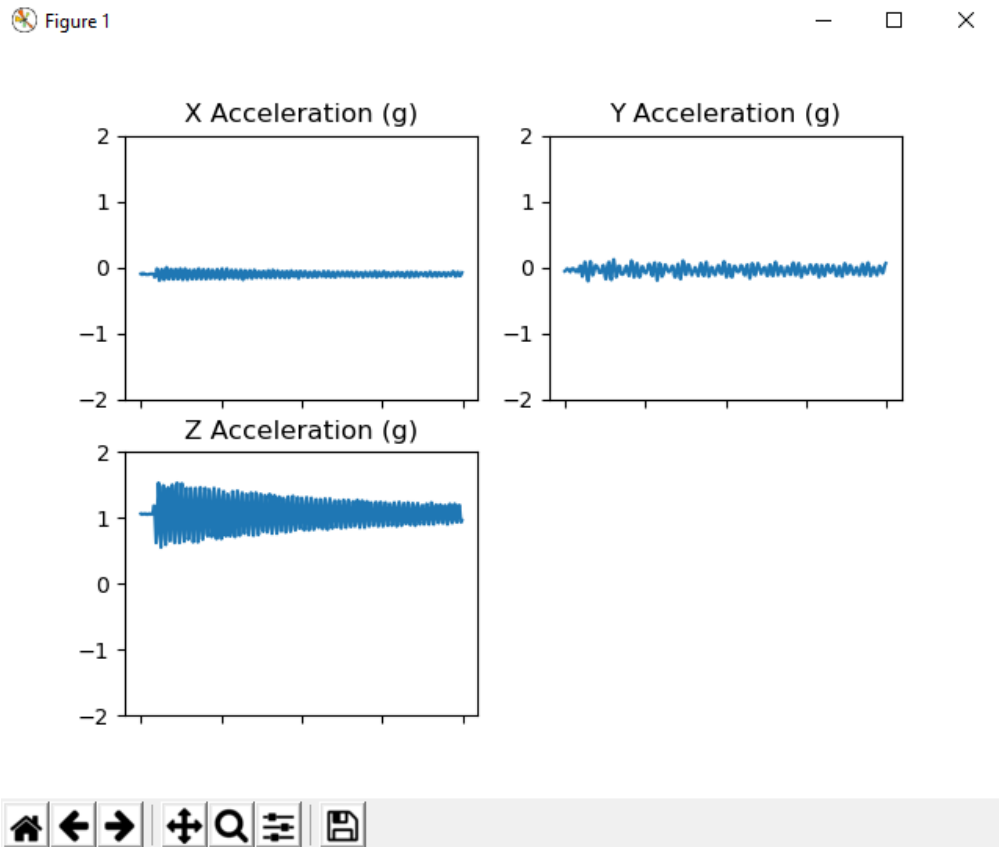
Αρχικά, ξεκινά μια ατέρμονη διαδικασία όπου μέσω μια βιβλιοθήκης (serial library) γίνεται ανάγνωση των αποτελεσμάτων που τυπώνονται σειριακά από το πρόγραμμα Arduino. Τι στιγμή εκείνη το πρόγραμμα αντιλαμβάνεται μια σειρά από τυχαίους χαρακτήρες. Η αποκωδικοποίησή τους στις τιμές των επιταχύνσεων όπως αυτές αντιστοιχούν στους άξονες X, Y, Z σε κάθε χρονικό βήμα γίνεται μέσω μιας βιβλιοθήκης (re library) αναγνώρισης χαρακτήρων. Η γνώση της θέσης όπου τυπώνονται οι διάφορες τιμές σειριακά δίνει τη δυνατότητα να μπορούν να εντοπιστούν μέσα στη σειρά από χαρακτήρες. Οι τιμές αυτές, καθώς και το χρονικό βήμα στο οποίο μετρήθηκαν, αποθηκεύονται σε ένα αρχείο τύπου txt ώστε ο χρήστης να μπορεί να τις επεξεργαστεί αργότερα (Εικόνα 2.9). Ταυτόχρονα, όλες οι μετρήσεις παρουσιάζονται σε πραγματικό χρόνο σε ένα ξεχωριστό παράθυρο γραφημάτων ώστε να υπάρχει μια εποπτεία της εξεταζόμενης ταλάντωσης (Εικόνα 2.10). Για τα γραφήματα χρησιμοποιήθηκε το matplotlib, μια βιβλιοθήκη της Python σχεδιασμένη για τη δημιουργία γραφημάτων. Μόλις ο χρήστης κρίνει ότι οι μετρήσεις που έχει πάρει είναι αρκετές ή ότι η εξεταζόμενη ταλάντωση έχει ολοκληρωθεί, κλείνοντας το παράθυρο των επιταχύνσεων διακόπτει την αποθήκευση των καταγραφών και ο αλγόριθμος βγαίνει από το βρόγχο.

accelerations_data1 - Notepad

File Edit Format View Help

Time(millisec)	AccX(g)	AccY(g)	AccZ(g)
0	-0.1	-0.04	1.05
5.0	-0.11	-0.04	1.06
9.0	-0.11	-0.04	1.05
14.0	-0.1	-0.04	1.05
18.0	-0.11	-0.04	1.06
23.0	-0.11	-0.05	1.05
27.0	-0.1	-0.04	1.06
31.0	-0.1	-0.04	1.05
36.0	-0.11	-0.04	1.05
40.0	-0.1	-0.04	1.06
45.0	-0.1	-0.04	1.06
49.0	-0.11	-0.04	1.06
54.0	-0.1	-0.04	1.05
58.0	-0.1	-0.04	1.06
62.0	-0.11	-0.04	1.06
67.0	-0.1	-0.04	1.06
71.0	-0.1	-0.04	1.04
76.0	-0.11	-0.04	1.06
80.0	-0.1	-0.05	1.05
85.0	-0.1	-0.05	1.06
89.0	-0.11	-0.05	1.06
94.0	-0.11	-0.04	1.05
98.0	-0.1	-0.04	1.05
102.0	-0.1	-0.05	1.05
107.0	-0.1	-0.04	1.05
111.0	-0.11	-0.04	1.05

Εικόνα 2.9: Αποθήκευση των αποτελεσμάτων σε αρχείο txt.



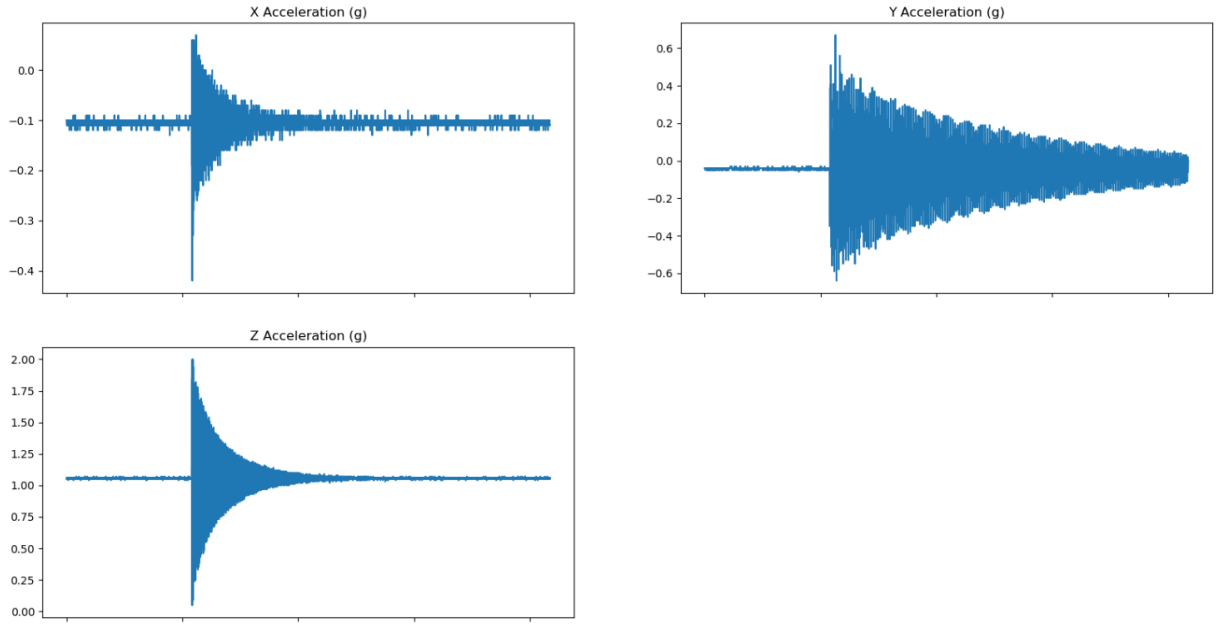
Εικόνα 2.10: Παράθυρο παρουσίασης των επιταχύνσεων στους 3 άξονες (X, Y, Z) σε πραγματικό χρόνο μέσω της βιβλιοθήκης matplotlib

Στη συνέχεια, τυπώνονται κάποιες επιπλέον πληροφορίες σχετικές με τις καταγραφές στο παράθυρο των εξαγόμενων αποτελεσμάτων του PyCharm. Αυτές είναι η διάρκεια της καταγραφής σε δευτερόλεπτα, το πλήθος των τιμών που λήφθηκαν, η συχνότητα της καταγραφής σε Hz καθώς και η μέγιστη επιτάχυνση που αναπτύχθηκε σε κάθε άξονα κατά τη διάρκεια της ταλάντωσης (Εικόνα 2.11).

Ακολουθούν τα διαγράμματα των επιταχύνσεων ολόκληρης της χρονοϊστορίας ώστε να μπορεί να γίνει εμφανές σε ποιους άξονες πραγματοποιήθηκε ταλάντωση και σε ποιους όχι (Εικόνα 2.12). Σε κάθε διάγραμμα το εύρος των τιμών των επιταχύνσεων είναι μεταβλητό και ανάλογο των καταγραφών. Έπειτα, υπολογίζεται για κάθε διεύθυνση η κύρια συχνότητα ταλάντωσης μέσα από μια διαδικασία που θα αναλυθεί σε επόμενη ενότητα και τα αποτελέσματα της παρουσιάζονται στον χρήστη (Εικόνα 2.13).

```
Total number of samples:  3658
Duration:  16.27 sec
Sampling frequency:  225 Hz
Maximum acceleration in X axis:  0.04  g
Maximum acceleration in Y axis:  0.39  g
Maximum acceleration in Z axis:  1.6  g
```

Εικόνα 2.11: Παρουσίαση των αποτελεσμάτων για τη δεδομένη καταγραφή όπως παρουσιάζονται στον χρήστη

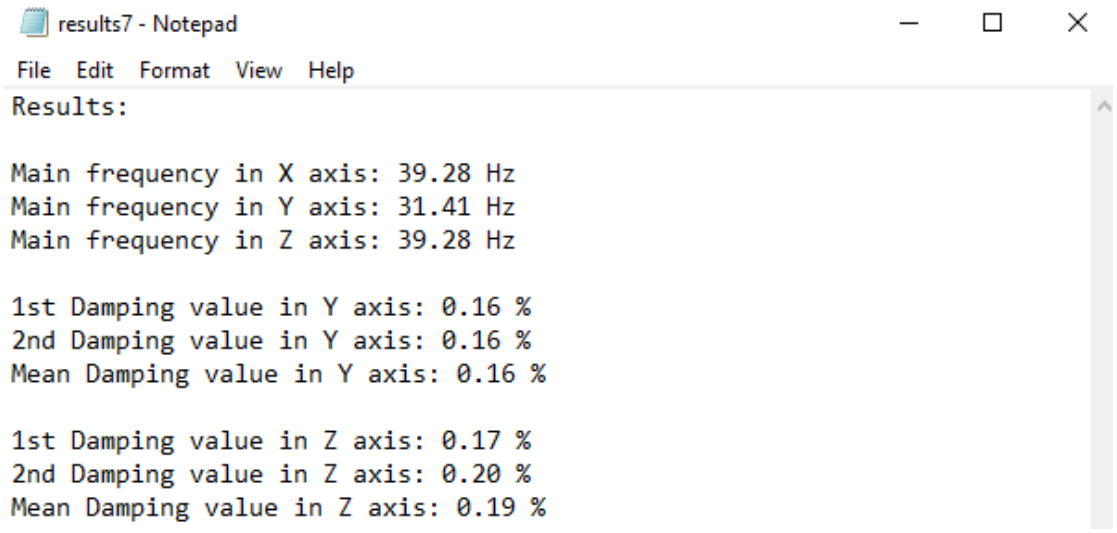


Εικόνα 2.12: Γράφημα ολόκληρης της χρονοϊστορίας των καταγραφών στους άξονες X, Y, Z

Ο χρήστης έχοντας την εικόνα της χρονοϊστορίας των επιταχύνσεων κατά τη διάρκεια της καταγραφής, όπως αυτή παρουσιάστηκε στα προηγούμενα διαγράμματα, καλείται να κρίνει σε ποιους άξονες θέλει να υπολογιστεί ο συντελεστής απόσβεσης, καθώς υπάρχει περίπτωση σε κάποιον άξονα να μην εκτελέστηκε κάποια μορφή ταλάντωσης (Εικόνα 2.13). Η επιλογή των αξόνων για τους οποίους θα υπολογιστεί ο συντελεστής απόσβεσης γίνεται πληκτρολογώντας τον αριθμό «1» στο παράθυρο εξόδου της Python. Ο αλγόριθμος είναι έτσι σχεδιασμένος ώστε να μπορεί να δώσει σωστά αποτελέσματα μόνο όταν πραγματοποιείται ταλάντωση σε όλους του άξονες επομένως το συγκεκριμένο βήμα είναι απαραίτητο για την αποφυγή οποιασδήποτε δυσλειτουργίας. Για τους άξονες που ζητήθηκε ο συντελεστής απόσβεσης τυπώνονται δύο τιμές, αλλά και η μέση τιμή αυτών σε ποσοστό % (Εικόνα 2.13). Τέλος, οποιοδήποτε αποτέλεσμα προέκυψε από τη καταγραφή αυτή, αποθηκεύεται σε ένα αρχείο txt ώστε να μπορεί ο χρήστης να ανατρέξει σε αυτό ακόμα και μετά την εκτέλεση του προγράμματος (Εικόνα 2.14).

```
    Main frequency in X axis:  37.76  Hz
    Main frequency in Y axis:  30.35  Hz
    Main frequency in Z axis:  37.76  Hz

Type 1 to calculate damping in X axis:00
Type 1 to calculate damping in Y axis:1
Type 1 to calculate damping in Z axis:1

1st Damping value in Y axis:  0.04  %
2nd Damping value in Y axis:  0.05  %
Mean damping value in Y axis: 0.04  %

1st Damping value in Z axis:  0.30  %
2nd Damping value in Z axis:  0.34  %
Mean damping value in Z axis: 0.32  %

Process finished with exit code 0
```

Εικόνα 2.13: Αποτελέσματα μετρήσεων με τη μορφή output στη Python



Εικόνα 2.14: Αρχείο txt με ό,τι αποτέλεσμα παρουσιάζεται στον χρήστη κατά τη διάρκεια εκτέλεσης του αλγορίθμου.

2.5.2 Εύρεση συχνοτήτων ταλάντωσης μέσα από επιταχυνσιογραφήματα

Ο υπολογισμός των συχνοτήτων ταλάντωσης όπως αυτές προκύπτουν από τα επιταχυνσιογραφήματα επιλέχτηκε να γίνεται με χρήση του μετασχηματισμού Fourier, μιας μεθοδολογίας που χρησιμοποιείται αρκετά συχνά για την ανάλυση σημάτων. Ο Joseph Fourier ήταν Γάλλος μαθηματικός και φυσικός που έμεινε στην ιστορία για την ενασχόλησή του με θέματα που αφορούν την μετάδοση της θερμότητας και τις ταλαντώσεις. Ο Fourier ανακάλυψε ότι τα επαναλαμβανόμενα κύματα μπορούν να αναπαρασταθούν ως ένα άθροισμα απείρων αρμονικών ημιτονοειδών κυμάτων (Σχήμα

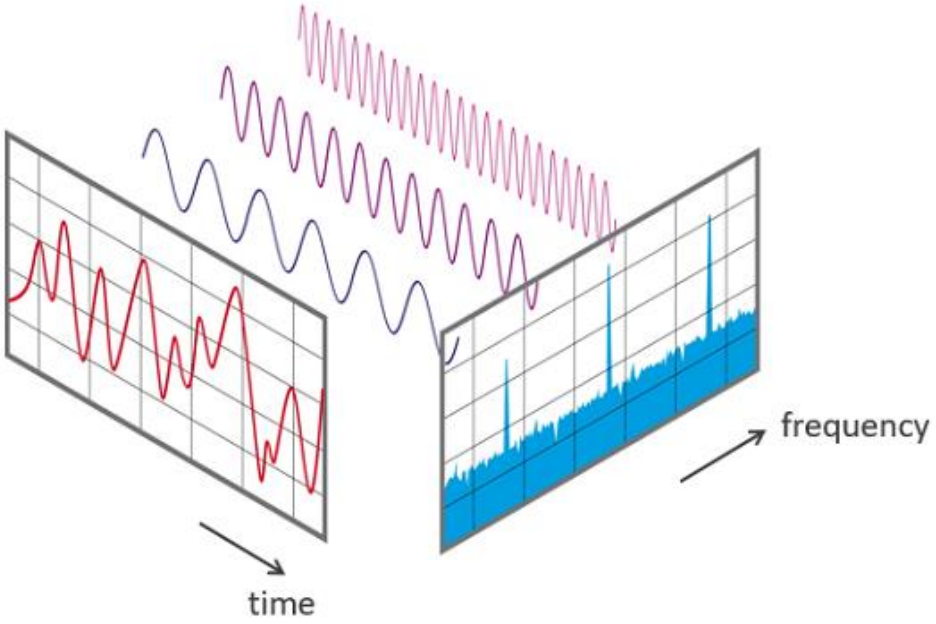
2.1). Η ανάλυση που φέρει το όνομά του είναι μια διαδικασία που προσπαθεί να εντοπίσει τις συχνότητες των κυμάτων που συνεισφέρουν στη διαμόρφωση του καταγραφέντος σήματος (σχέση (2.1)), όπου X(j) μια μεταβλητή στον χώρο των συχνοτήτων, j ο αριθμός μιας συχνότητας, x(k) μια μεταβλητή στον χώρο των πραγματικών τιμών και k ο αριθμός μια πραγματικής τιμής. Στην ουσία, η ανάλυση του Fourier παίρνει ένα κύμα οποιασδήποτε μορφής και το μετατρέπει σε ένα γράφημα συχνοτήτων όπου τα διάφορα μέγιστά του αντιπροσωπεύουν τις συχνότητες των αρμονικών κυμάτων που συμμετέχουν περισσότερο σε αυτό (Σχήμα 2.2).

$$X(j) = \int_{-\infty}^{+\infty} x(k) e^{-i2\pi jk} dx \text{, για j = 0, 1, …, N – 1 και k = 0, 1, …, N - 1} \quad (2.1)$$

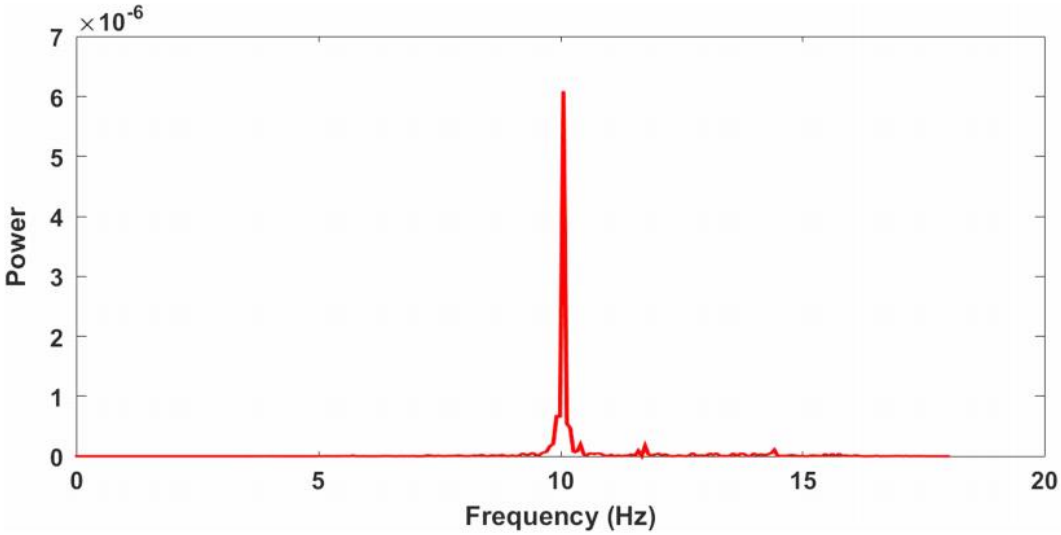
Ο διακριτός μετασχηματισμός Fourier (DFT) είναι μια μετεξέλιξη του μετασχηματισμού Fourier που για την ανάλυση σημάτων χρησιμοποιεί αθροίσματα αντί για ολοκληρώματα, όπως φαίνεται και από τη σχέση (2.2).

$$X(j) = \sum_{k=0}^{N-1} x(k) e^{-i2\pi jk/N} \text{ , για j = 0, 1, …, N – 1 και k = 0, 1, …, N - 1} \quad (2.2)$$

Ωστόσο, η ανάγκη για επίτευξη γρήγορων υπολογιστικών ταχυτήτων οδήγησε στην δημιουργία ενός αλγορίθμου με το όνομα γρήγορος μετασχηματισμός Fourier (FFT). Ο αλγόριθμος αυτός υλοποιεί τον διακριτό μετασχηματισμό Fourier σε ζεύγη τιμών και για αυτό απαιτεί το πλήθος των δειγμάτων να είναι δύναμη του 2. Βέβαια, οι ταχύτητες επεξεργασίας των δεδομένων σήμερα είναι αρκετά υψηλές με αποτέλεσμα τέτοιου είδους αλγόριθμοι να μην έχουν αισθητές διαφορές από τις παραδοσιακότερες μεθόδους, παρόλα αυτά θα χρησιμοποιηθεί στη παρούσα εργασία. Πριν την εκτέλεση του μετασχηματισμού θα πρέπει να είναι γνωστά δύο μεγέθη, η συχνότητα καταγραφής (sampling rate) και ο αριθμός των μετρήσεων (samples). Από τα δύο αυτά μεγέθη προκύπτουν πληροφορίες σχετικές με τα αποτελέσματα του μετασχηματισμού, όπως η ακρίβεια των αποτελεσμάτων και η μέγιστη συχνότητα που μπορεί να ανιχνευτεί. Η συχνότητα καταγραφής θα αποτελέσει αντικείμενο διερεύνησης της παρούσας εργασίας καθώς από αυτήν εξαρτάται η λεπτομέρεια με την οποία αποθηκεύεται ένα σήμα. Για παράδειγμα, ένα σήμα των 20 Hz που καταγράφεται με συχνότητα 100 Hz αναπαρίσταται από 5 σημεία (100/20=5), ενώ αν καταγράφεται με 200 Hz αναπαρίσταται από 10 σημεία (200/20=10). Γίνεται εύκολα αντιληπτό ότι όσο μεγαλύτερη είναι η συχνότητα καταγραφής τόσο μικρότερα σφάλματα θα υπάρχουν. Όμως από ποια συχνότητα και μετά παύουν να εμφανίζονται διαφορές στα αποτελέσματα των μετρήσεων;



Σχήμα 2.1: Αποσύνθεση ενός τυχαίου σήματος σε άθροισμα αρμονικών κυμάτων μέσω του μετασχηματισμού Fourier (από www.nti-audio.com)



Σχήμα 2.2: Διάγραμμα συχνότητας-ισχύος (φασμα ισχύος) όπως προκύπτει από τον μετασχηματισμό Fourier ενός τυχαίου κύματος (από Varanis M., Silva A.L., Brunetto P.H. και Gregolin R.F.)

2.5.3 Υπολογισμός συντελεστή απόσβεσης μέσα από επιταχυνσιογραφήματα

Ο αλγόριθμος που αναπτύχθηκε στη παρούσα εργασία δίνει τη δυνατότητα στον χρήστη να υπολογίσει τον συντελεστή απόσβεσης της ταλάντωσης που καταγράφεται. Ο συντελεστής απόσβεσης είναι ένα μέγεθος που δεν μπορεί να προσδιοριστεί μαθηματικά από τις φυσικές και τις γεωμετρικές ιδιότητες του φορέα, παρά μόνο μέσω του τρόπου με

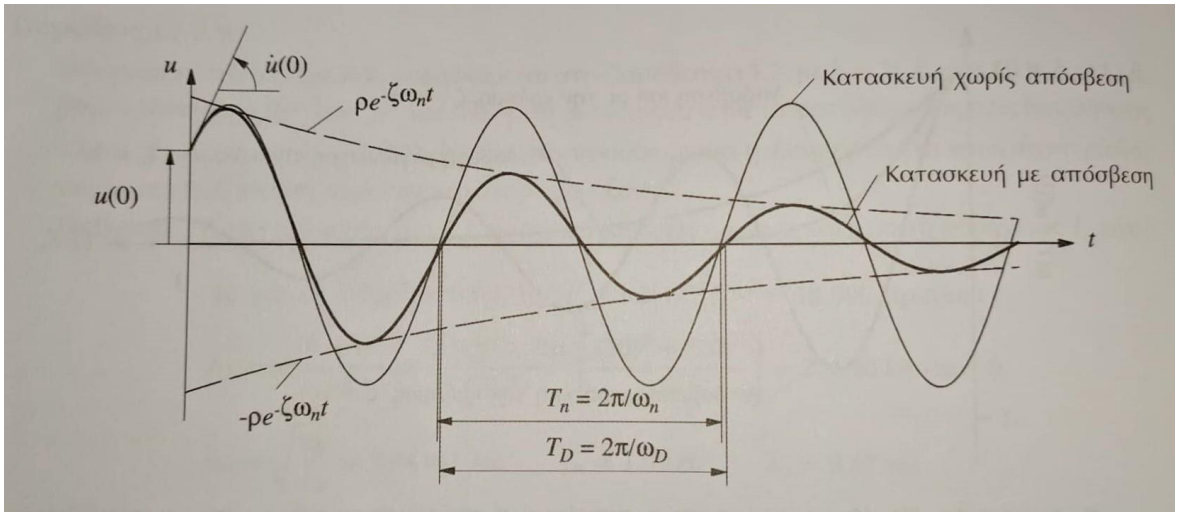
τον οποίο ταλαντώνεται. Για να γίνει αυτό θα πρέπει αν είναι γνωστή η χρονοϊστορία της ταλάντωσης του.

Έχει αποδειχθεί ότι εύρος της ταλάντωσης ενός μονοβάθμιου ταλαντωτή με απόσβεσης μειώνεται εκθετικά με τον χρόνο (Σχήμα 2.3) και πως οι περιβάλλουσες καμπύλες της χρονοϊστορίας του εκφράζονται μέσω της σχέσης $\pm\rho e^{-\zeta\omega_n t}$, όπου:

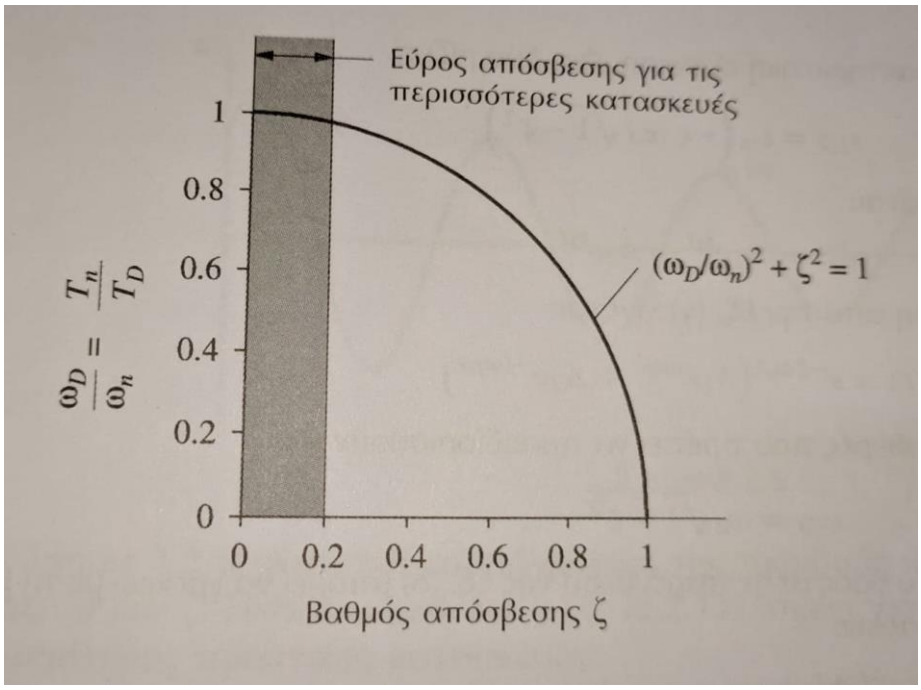
$$\rho = \sqrt{[u(0)]^2 + [\frac{\dot{u}(0) + \zeta\omega_n u(0)}{\omega_D}]^2} \tag{2.3}$$

το εύρος ταλάντωσης του φορέα τη χρονική στιγμή t = 0, u(0) η μετακίνησή του εκείνη τη χρονική στιγμή, $\dot{u}(0)$ η ταχύτητά του, ζ ο συντελεστής απόσβεσης, ω_n η ιδιοσυχνότητα του χωρίς απόσβεσης και ω_D η ισιοδυχνότητά με απόσβεση. Επιπλέον έχει αποδειχθεί ότι για αποσβέσεις μικρότερες του 20 % η ιδιοσυχνότητα του ταλαντωτή με απόσβεση είναι ακριβώς η ίδια με αυτήν που θα είχε αν δεν είχε απόσβεση (Σχήμα 2.4). Δεδομένου ότι σε συνήθεις κατασκευές πολιτικού μηχανικού η απόσβεση φτάνει το 5 % (κατασκευές από σκυρόδεμα), θεωρείται πως το ω_n ταυτίζεται με το ω_D.

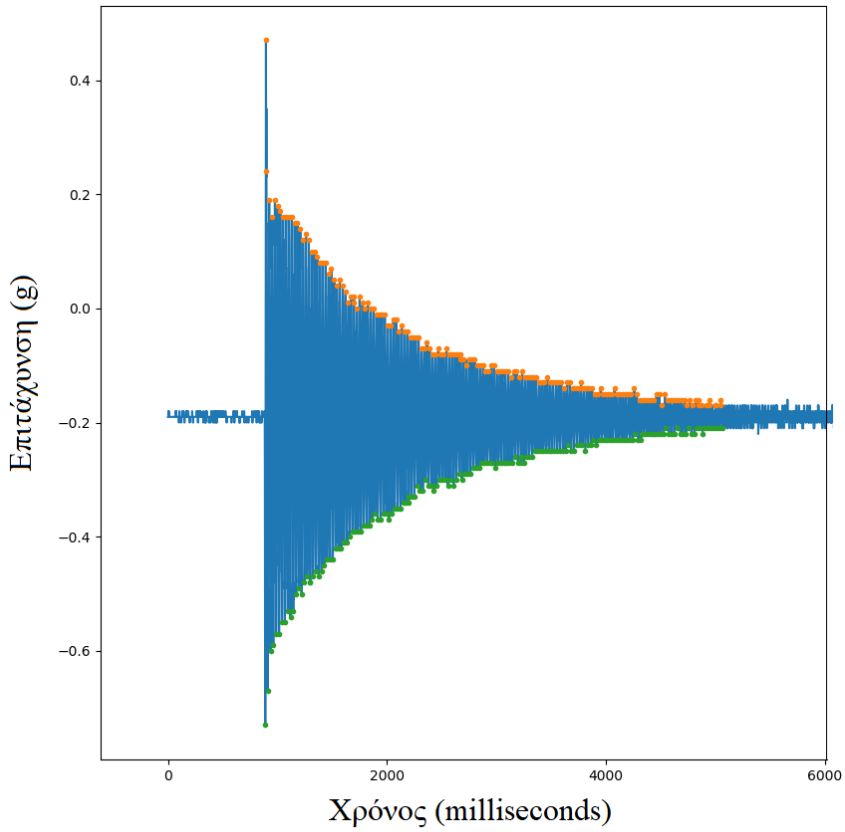
Για τον υπολογισμό του συντελεστή απόσβεσης όπως αυτός προκύπτει από κάθε καταγραφή αναπτύχθηκε ένας αλγόριθμος που εντοπίζει τα τοπικά μέγιστα και ελάχιστα του κατεγραφέντος επιταχυνσιογραφήματος (Σχήμα 2.5). Στη συνέχεια υπολογίζεται ο λογάριθμος των επιταχύνσεων που αντιστοιχούν στα τοπικά μέγιστα και ελάχιστα ώστε να προκύψει ένα διάγραμμα συναρτήσει του χρόνου. Η κλίση της ευθείας που δημιουργείται από τα ελάχιστα τετράγωνα αυτών των σημείων θα ισούται με τον συντελεστή απόσβεσης της συγκεκριμένης ταλάντωσης (Σχήμα 2.6). Όπως αναμένεται, επειδή υπάρχει ένα θετικός και ένας αρνητικός κλάδος ταλάντωσης προκύπτουν δύο τιμές του συντελεστή απόσβεσης και για αυτό στο τέλος του αλγορίθμου υπολογίζεται και μια μέση τιμή αυτών.



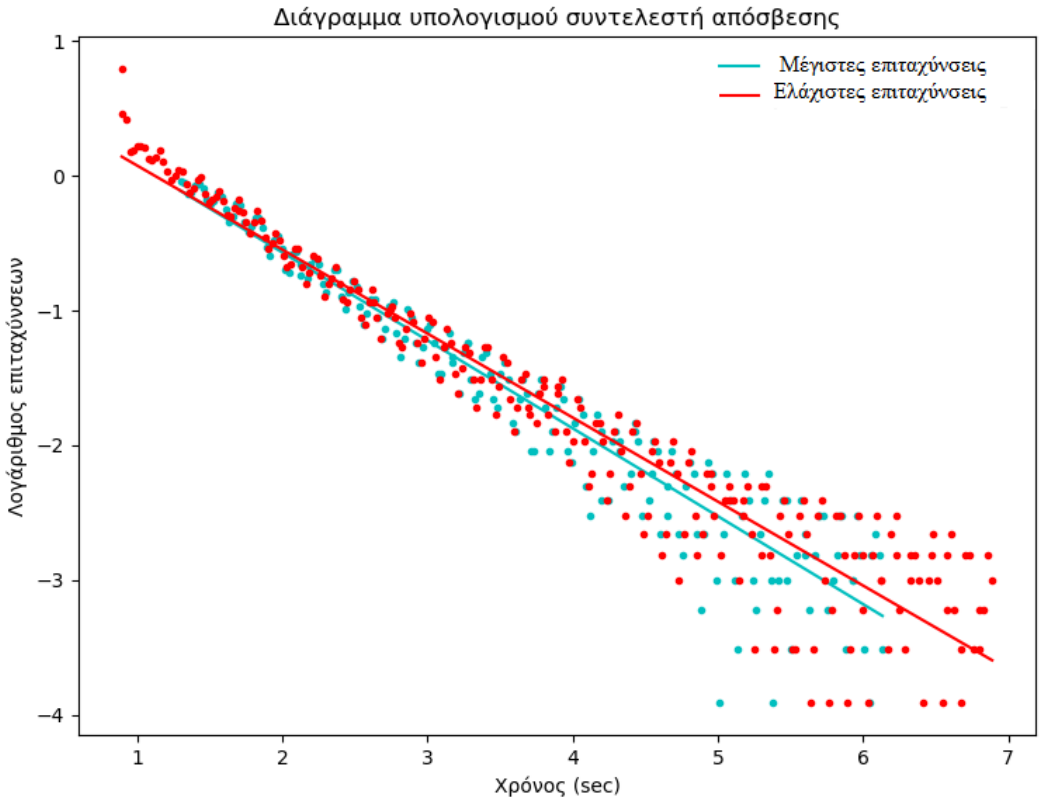
Σχήμα 2.3: Χρονοϊστορία ταλάντωσης φορέα με απόσβεση και εξίσωση περιβάλλουσας της (από Chopra A.K., 2018)



Σχήμα 2.4: Σχέση ιδιοσυχνότητας με και χωρίς απόσβεση συναρτήσει του συντελεστή απόσβεσης (από Chopra A.K., 2018)



Σχήμα 2.5: Εύρεση τοπικών μέγιστων (πορτοκαλί χρώμα) και ελάχιστων (πράσινο χρώμα) σε επιταχυνσιογράφημα



Σχήμα 2.6: Λογάριθμος τοπικών μέγιστων και ελάχιστων επιταχύνσεων συναρτήσει του χρόνου και χάραξη της βέλτιστης γραμμικής σχέσης με τη μέθοδο ελαχίστων τετραγώνων για υπολογισμό του συντελεστή απόσβεσης

2.5.4 Παραλλαγές αλγορίθμου

Η ανάγκη για παρουσίαση των τιμών που καταγράφονται σε πραγματικό χρόνο, όπως θα φανεί και σε μετέπειτα δοκιμές, μπορεί να οδηγήσει σε σημαντικά σφάλματα. Αυτό συμβαίνει καθώς απαιτείται σημαντικό μέρος της μνήμης RAM για την εκτέλεση μιας τέτοιας διεργασίας. Παρατηρήθηκε ότι υπήρχαν περιπτώσεις που κατά την διάρκεια μιας καταγραφής εμφανιζόταν ένα χρονικό κενό μεταξύ των μετρήσεων, το οποίο μπορεί να έχει αρνητικές επιπτώσεις για τα εξαγόμενα αποτελέσματα. Για τον λόγο αυτό, αναπτύχθηκαν άλλοι δύο αλγόριθμοι με γνώμονα τον αρχικό. Στόχος ήταν να μειωθεί ο υπολογιστικός φόρτος του αλγορίθμου και ταυτοχρόνως να αυξηθεί η συχνότητα καταγραφής των μετρήσεων κάτω από ένα σταθερό χρονικό βήμα.

Κατά τη δεύτερη εκδοχή που αναπτύχθηκε οι τροποποιήσεις ήταν ελάχιστες. Η κύρια αλλαγή ήταν να απαλλαγεί ο αλγόριθμος από την παρουσίαση των μετρούμενων τιμών σε πραγματικό χρόνο. Ο αλγόριθμος μπορεί να διαβάζει τις μετρήσεις από το πρόγραμμα του Arduino χωρίς όμως να τις παρουσιάζει στον χρήστη εκείνη τη στιγμή. Η παρουσίασή τους γίνεται μόλις διακοπεί η καταγραφή των τιμών με το πάτημα ενός κουμπιού. Η τροποποίηση αυτή οδήγησε σε αυξημένη συχνότητα καταγραφής αλλά και σε σταθερό χρονικό βήμα, απαραίτητο για την σωστή εκτέλεση του μετασχηματισμού Fourier. Στη συνέχεια, η επεξεργασία και η εκτύπωση των αποτελεσμάτων είναι ακριβώς ίδια με αυτή του αρχικού αλγορίθμου.

Για την τρίτη και τελευταία εκδοχή αποφασίστηκε να μην υπάρχει άμεση επικοινωνία μεταξύ Arduino και Python. Για να επιτευχθεί κάτι τέτοιο αναπτύχθηκε ένας αλγόριθμος ικανός να διαβάζει μετρήσεις αποθηκευμένες σε αρχεία txt. Με τον τρόπο αυτό ήταν εφικτό να μειωθεί σημαντικά το υπολογιστικό κόστος του προγράμματος. Στην προσπάθεια να μειωθεί και άλλο ο υπολογιστικός φόρτος και ταυτοχρόνως να αυξηθεί η συχνότητα καταγραφής των επιταχύνσεων, τροποποιήθηκε και ο κώδικας λειτουργίας του μικροεπεξεργαστή. Παρατηρήθηκε πως λίγο της εκτύπωσης των τιμών των επιταχύνσεων στο σειριακό παράθυρο (serial monitor) του Arduino, υπήρχε ένα ανώτερο όριο στη ταχύτητα λειτουργίας του αλγορίθμου. Για να αυξηθεί αυτό το όριο χρειάστηκε οι μετρήσεις να αποθηκεύονται σε λίστες μεταβλητών και να τυπώνονται μετά το πέρας της καταγραφής. Για να γίνει αυτό έπρεπε να έχει οριστεί από πριν το μέγεθος των μεταβλητών αυτών. Δεδομένου ότι ο μικροεπεξεργαστής Teensy 3.6 έχει 256 kilobytes μνήμης RAM και οι μεταβλητές είναι τέσσερις (χρόνος και τρεις άξονες επιτάχυνσης) υπήρχε δυνατότητα για καταγραφή 15.000 τιμών για κάθε μεταβλητή. Ο συγκεκριμένος αριθμός τιμών οδήγησε σε ποσοστό εκμετάλλευσης της μνήμης του Teensy 94 %, ποσοστό οριακό για την ομαλή εκτέλεση των απαιτούμενων λειτουργιών του. Με την παραπάνω τροποποίηση επιτεύχθηκε ταχύτητα καταγραφής μετρήσεων έως και 1200 Hz, παρόλα αυτά επιλέχτηκε οι όποιες καταγραφές έγινες με την συγκεκριμένη εκδοχή του αλγορίθμου να γίνουν με συχνότητα 1000 Hz. Δεδομένου ότι μπορούν να αποθηκευτούν μόνο 15000 τιμές από κάθε μεταβλητή και για συχνότητα καταγραφής 1000 Hz, γίνεται αντιληπτό ότι ο αλγόριθμος μπορεί να καταγράψει μέχρι και 15 δευτερόλεπτα ταλάντωσης. Κάτι τέτοιο ίσως αποτελεί και το μειονέκτημα μια υψίσυχνης με τα συγκεκριμένα εργαλεία και αλγόριθμο.

3 Αναγνώριση κυρίων ιδιοσυχνοτήτων ταλάντωσης και συντελεστών απόσβεσης μεταλλικού ελάσματος

3.1 Εισαγωγή

Η πρώτη δοκιμή του αισθητήρα και του κώδικα με τον οποίο γίνεται η επεξεργασία των αποτελεσμάτων επιλέχτηκε να γίνει σε ένα μεταλλικό χάρακα όπου το ένα το άκρο παρέμενε ακλόνητο (πακτωμένο). Η συγκεκριμένη επιλογή έγινε λόγω της ευκολίας μεταβολής των ιδιομορφικών χαρακτηριστικών του φορέα και κατά συνέπεια της δυνατότητας παρακολούθησης της απόκρισης του αλγορίθμου σε ταλαντώσεις διαφορετικού εύρους συχνοτήτων.

3.2 Περιγραφή φορέα

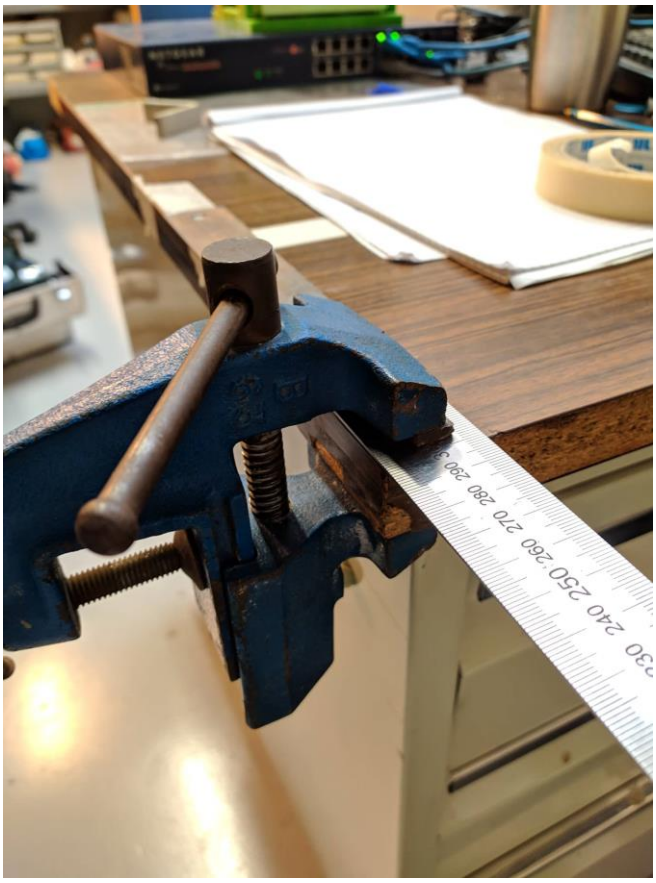
Ο χάρακας που επιλέχτηκε είναι χαλύβδινος συνολικού μήκους 500 mm και η διατομή του έχει ύψος h 1 mm και πλάτος b 30.2 mm. Ο χάρακας στηρίχθηκε σε ένα γραφείο μέσω ενός σφιγκτήρα (Εικόνα 3.1) δημιουργώντας συνθήκες προβόλου καθώς με αυτόν τον τρόπο το ένα του άκρο παρέμενε πάντα ακλόνητο και το άλλο ήταν ελεύθερο να ταλαντωθεί (Εικόνα 3.2). Στη προσπάθεια να διερευνηθούν διαφορετικές συχνότητες ταλάντωσης, έγιναν μετρήσεις σε 6 διαφορετικά ελεύθερα μήκη. Τα μήκη L που εξετάστηκαν ήταν τα 150, 200, 250, 300, 350 και 400 mm. Δεδομένου ότι δεν άλλαζε κανένα χαρακτηριστικό του φορέα παρά μόνο το μήκος του, μεταβολή του μήκους του φορέα ισοδυναμούσε με αύξηση ή μείωση της δυσκαμψίας του και ακολούθως της συχνότητας ταλάντωσής του. Για τον υπολογισμό της συχνότητας ταλάντωσης του χάρακα λήφθηκε υπόψη και η μάζα m_{acc} του αισθητήρα και των καλωδίων καθώς ήταν συγκρίσιμη με αυτή του χάρακα m. Η μάζα αυτή θεωρήθηκε ως συγκεντρωμένη στο ελεύθερο άκρο του φορέα και μετρήθηκε στα 32 gr (Εικόνα 3.3). Ως μάζα M ορίζεται η ταλαντευόμενη που αντιστοιχεί στην πρώτη και κύρια ιδιομορφή. Το μέτρο ελαστικότητας του χάλυβα E θεωρήθηκε 206.8 GPa και η πυκνότητά του 7850 kg/m^3. Ο υπολογισμός των θεωρητικών τιμών των ιδιοσυχνοτήτων ταλάντωσης f που αντιστοιχούν στα διάφορα μήκη L έγινε μέσω των σχέσεων (3.1), (3.2), (3.3) και (3.4), οι οποίες προκύπτουν από την επίλυση της εξίσωσης κίνησης προβόλου για την κύρια ιδιοσυχνότητα του. Στον Πίνακα 3.1 παρουσιάζονται οι τιμές αυτές για τα εξεταζόμενα μήκη.

$$f = \frac{1}{2\pi}\sqrt{\frac{k}{M}}, \tag{3.1}$$

$$k = \frac{3EI}{L^3}, \tag{3.2}$$

$$I = \frac{bh^3}{12}, \tag{3.3}$$

$$M = \frac{33}{140}m + m_{acc}, \tag{3.4}$$

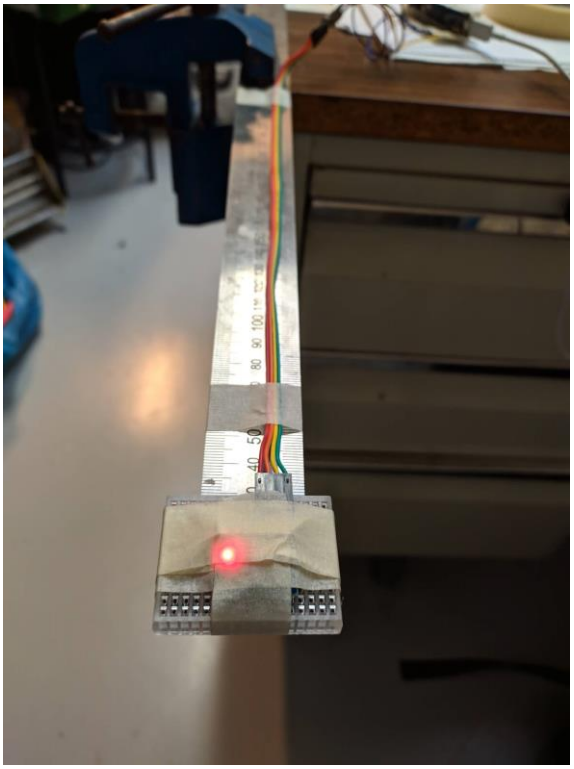


Εικόνα 3.1: Στήριξη του ελάσματος στο ένα του άκρο με σφιγκτήρα

Εικόνα 3.2: Προσομοίωση προβόλου μέσω ελάσματος

Πίνακας 3.1: Υπολογισμός δυσκαμψίας, μάζας ελάσματος, ταλαντευόμενης μάζας και συχνότητας ταλάντωσης των εξεταστέων μηκών προβόλου.

L (mm)	k (kN/m)	m (kg)	M (kg)	f (Hz)
150	0.4626	0.0356	0.0254	21.49
200	0.1952	0.0474	0.0282	13.25
250	0.0999	0.0593	0.0310	9.04
300	0.0578	0.0711	0.0338	6.59
350	0.0364	0.0830	0.0366	5.02
400	0.0244	0.0948	0.0394	3.96



Εικόνα 3.3: Τοποθέτηση αισθητήρα στο ελεύθερο άκρο του ελάσματος

3.3 Δεδομένα μετρήσεων και αποτελέσματα

Για κάθε ελεύθερο μήκος χάρακα έγιναν πέντε δειγματοληψίες από τις οποίες προέκυψαν οι μετρούμενες συχνότητες ταλάντωσης σε Hz, το σφάλμα σε σχέση με την αντίστοιχη θεωρητική τιμή και η απόσβεση σε ποσοστό %. Για τις τιμές αυτές υπολογίστηκε και μια μέση τιμή. *Η καταγραφή των επιταχύνσεων σε πρώτη φάση έγινε στα 204 Hz*, δηλαδή ανά δευτερόλεπτο λαμβάνονταν 204 τιμές επιτάχυνσης περί τον άξονα ταλάντωσης του χάρακα. Η συχνότητα αυτή είναι η μέγιστη δυνατή που μπορούσε να επιτευχθεί. Τα αποτελέσματα των μετρήσεων παρουσιάζονται στον Πίνακα 3.2.

Πίνακας 3.2: Αποτελέσματα μετρήσεων συχνότητας και απόσβεσης καθώς και απόκλιση των πρώτων από τη θεωρητική τιμή για μεταβλητό μήκος χάρακα-προβόλου και σταθερή συχνότητα καταγραφής στα 204 Hz.

Μήκος (mm)	Θεωρητική συχνότητα (Hz)	Μετρούμενη συχνότητα (Hz)	Απόκλιση συχνότητας (%)	Μετρούμενη Απόσβεση (%)
150	21.49	19.38	9.8	0.43
		19.65	8.5	0.37
		19.67	8.5	0.38
		19.42	9.6	0.38
		19.7	8.3	0.36
	Μέση τιμή	*19.56*	*8.9*	*0.38*
200	13.25	12.43	6.2	0.45
		12.57	5.1	0.45
		12.6	4.9	0.43
		12.44	6.1	0.47
		12.58	5.0	0.39
	Μέση τιμή	*12.52*	*5.5*	*0.44*
250	9.04	8.63	4.5	0.35
		8.68	4.0	0.33
		8.79	2.8	0.24
		8.76	3.1	0.35
		8.77	3.0	0.33
	Μέση τιμή	*8.73*	*3.5*	*0.32*
300	6.59	6.39	3.0	0.22
		6.41	2.7	0.22
		6.41	2.7	0.22
		6.37	3.3	0.23
		6.37	3.3	0.23
	Μέση τιμή	*6.39*	*3.0*	*0.22*
350	5.02	4.88	2.8	0.21
		4.85	3.4	0.22
		4.86	3.2	0.22
		4.89	2.7	0.19
		4.88	2.8	0.21
	Μέση τιμή	*4.87*	*3.0*	*0.21*
400	3.96	3.80	4.1	0.13
		3.85	2.8	0.14
		3.86	2.6	0.13
		3.84	3.1	0.13
		3.84	3.1	0.11
	Μέση τιμή	*3.84*	*3.1*	*0.13*

Παρατηρώντας τα αποτελέσματα προκύπτει ότι στις περισσότερες των περιπτώσεων ο συντελεστής απόσβεσης αυξάνεται όσο αυξάνεται η δυσκαμψία του προβόλου. Επίσης, παρατηρείται ότι για τα πιο δύσκαμπτα μήκη η απόκλιση των συχνοτήτων από την αντίστοιχη θεωρητική είναι μεγάλη, της τάξης του 5 με 9 %. Το γεγονός αυτό οδήγησε στην ανάγκη μιας περεταίρω διερεύνησης σχετικά με την συχνότητα με την οποία θα πρέπει να καταγράφονται οι τιμές των επιταχύνσεων. Για τον λόγο αυτό, συγκρίνονται 12 διαφορετικές συχνότητες από τα 47 Hz μέχρι τα 220 Hz, με σταθερό μήκος χάρακα και ίσο με 250 mm. Τα αποτελέσματα αυτά παρουσιάζονται στον Πίνακα 3.3.

Πίνακας 3.3: Αποτελέσματα μετρήσεων για σταθερό μήκος χάρακα-προβόλου και μεταβλητή συχνότητα καταγραφής.

Συχνότητα καταγραφής (Hz)	Θεωρητική συχνότητα (Hz)	Μετρούμενη συχνότητα (Hz)	Απόκλιση Συχνότητας (%)	Απόσβεση (%)
204	9.04	8.63	4.5	0.24
184	9.04	8.9	1.6	0.25
155	9.04	8.91	1.4	0.24
134	9.04	8.89	1.7	0.25
118	9.04	8.88	1.8	0.27
106	9.04	8.92	1.3	0.25
96	9.04	8.92	1.3	0.25
87	9.04	8.85	2.1	0.29
80	9.04	8.91	1.4	0.24
74	9.04	8.89	1.7	0.28
65	9.04	8.89	1.7	0.25
47	9.04	8.89	1.7	0.27

Από τα παραπάνω αποτελέσματα εντοπίζονται τα εξής σημεία, ο συντελεστής απόσβεσης είναι ανεξάρτητος της συχνότητας καταγραφής, ενώ το ίδιο ισχύει και για τις συχνότητες ταλάντωσης με εξαίρεση αυτή των 220 Hz της οποίας η τιμή διαφέρει από την αντίστοιχη θεωρητική κατά 4.5 % στη δεδομένη καταγραφή. Πιθανή αιτία αυτής της παρατήρησης είναι η εξάντληση της υπολογιστικής δύναμης του συστήματος με το οποίο έγιναν οι καταγραφές αυτές. Στη προσπάθεια να επιτευχθεί η μέγιστη δυνατή συχνότητα καταγραφής, το χρονικό βήμα της δειγματοληψίας δεν είναι σταθερό, όπως θα έπρεπε, με αποτέλεσμα την εισαγωγή σφάλματος στον μετασχηματισμό Fourier που επιτελείται μέσα στον αλγόριθμο για την εξαγωγή των συχνοτήτων ταλάντωσης. Για τον λόγο αυτό, κρίθηκε και ασφαλέστερο να επαναληφθούν οι καταγραφές σε αυτό το μοντέλο με την αμέσως ταχύτερη συχνότητα καταγραφής, δηλαδή αυτή των 184 Hz δεδομένου ότι τα αποτελέσματα της διερεύνησης έδειξαν ότι μπορεί να δώσει μικρότερα σφάλματα. Στον Πίνακα 3.4 παρουσιάζονται οι μετρήσεις για τα έξι μήκη χάρακα *με συχνότητα καταγραφής τα 184 Hz.*

Πίνακας 3.4: Αποτελέσματα μετρήσεων για μεταβλητό μήκος χάρακα-προβόλου και σταθερή συχνότητα καταγραφής στα 184 Hz.

Μήκος (mm)	Θεωρητική συχνότητα (Hz)	Μετρούμενη συχνότητα (Hz)	Απόκλιση Συχνότητας (%)	Απόσβεση (%)
150	21.49	21.41	0.4	0.24
		21.46	0.1	0.22
		21.41	0.4	0.24
		21.45	0.2	0.21
		21.48	0.0	0.25
	Μέση τιμή	*21.44*	*0.2*	*0.23*
200	13.25	13.28	0.3	0.37
		13.27	0.2	0.31
		13.31	0.5	0.28
		13.26	0.1	0.26
		13.36	0.9	0.3
	Μέση τιμή	*13.30*	*0.4*	*0.30*
250	9.04	9.05	0.1	0.23
		9.07	0.3	0.22
		9.07	0.3	0.23
		9.04	0.0	0.25
		9.08	0.4	0.21
	Μέση τιμή	*9.06*	*0.2*	*0.23*
300	6.59	6.52	1.0	0.18
		6.51	1.2	0.15
		6.52	1.0	0.14
		6.53	0.9	0.15
		6.53	0.9	0.15
	Μέση τιμή	*6.52*	*1.0*	*0.15*
350	5.02	5.01	0.3	0.16
		5.00	0.5	0.16
		5.01	0.3	0.15
		4.98	0.9	0.14
		5.01	0.3	0.16
	Μέση τιμή	*5.00*	*0.4*	*0.15*
400	3.96	3.97	0.2	0.18
		3.96	0.1	0.20
		3.97	0.2	0.16
		3.95	0.3	0.20
		3.96	0.1	0.18
	Μέση τιμή	*3.96*	*0.2*	*0.18*

3.4 Συμπεράσματα

Η συχνότητα με την οποία καταγράφονται οι επιταχύνσεις παίζει σημαντικό ρόλο στην ακρίβεια των αποτελεσμάτων. Παρά την αοριστία που μπορεί να χαρακτηρίζει δεδομένα όπως το ακριβές βάρος του χάλυβα ή το μέτρο ελαστικότητάς του, η επίτευξη ενός σταθερού βήματος δειγματοληψίας είναι απαραίτητη για την ορθότητα των αποτελεσμάτων. Η γνώση αυτή θα αποτελέσει χρήσιμο εργαλείο για μετέπειτα δοκιμές σε συνθετότερους και ρεαλιστικότερους φορείς.

4 Αναγνώριση κύριας ιδιοσυχνότητας και συντελεστή απόσβεσης ταλάντωσης μεταλλικού προβόλου

4.1 Εισαγωγή

Η δεύτερη εφαρμογή του αισθητήρα και του λογισμικού που αναπτύχθηκε για τη επεξεργασία των μετρήσεων του έγινε σε μια μονοπροέχουσα δοκό (Εικόνα 4.1 και Εικόνα 4.2) στο χώρου του Εργαστηρίου Μεταλλικών Κατασκευών του Εθνικού Μετσόβιου Πολυτεχνείου. Επιλέχτηκε ο συγκεκριμένος φορέας καθώς κρίθηκε ότι ήταν αρκετά πιο ρεαλιστικός και κοντά σε εφαρμογές πολιτικού μηχανικού, χωρίς όμως να ξεφεύγει αρκετά από την λογική της πρώτης εφαρμογής.

4.2 Από τη θεωρία στη πράξη

Πριν την πραγματοποίηση οποιασδήποτε μέτρησης, εντοπίστηκαν τα γεωμετρικά χαρακτηριστικά της δοκού ώστε μέσα από αυτά να προκύψει η πρώτη συχνότητα ταλάντωσης με την οποία και θα συγκρίνονται τα αποτελέσματα του επιταχυνσιομέτρου. Αποφασίστηκε να εξεταστεί η ταλάντωση της δοκού μόνο περί τον ισχυρό άξονά της, δηλαδή για κίνηση στο κατακόρυφο επίπεδό της.

Με βάση τις πληροφορίες που δόθηκαν από το εργαστήριο, η δοκός είναι πρότυπης διατομής IPE 160, ενώ το μήκος της μετρήθηκε στα 2040 mm. Λαμβάνοντας το μέτρο ελαστικότητας του χάλυβα Ε στα 206,8 GPa και τη πυκνότητά του στα 7850 kg/m^3, υπολογίζονται τα αδρανειακά χαρακτηριστικά της δοκού (Πίνακας 4.1). Για τον υπολογισμό της πρώτης συχνότητας ταλάντωσης χρησιμοποιήθηκαν οι σχέσεις (3.1), (3.2), (3.3) και (3.4) οι οποίες θεωρούν την στήριξη στο άκρο του προβόλου ως πλήρη πάκτωση. Επιπλέον, το βάρος του επιταχυνσιομέτρου θεωρήθηκε αμελητέο συγκριτικά με το βάρος της δοκού.

Πίνακας 4.1: Αδρανειακά χαρακτηριστικά της υπό εξέταση δοκού

Εμβαδό (A)	20.1	cm^2
Μήκος (L)	2040.0	mm
Ροπή Αδρανείας (I)	869	cm^4
Μέτρο Ελαστικότητας (E)	206.8	GPa
Πυκνότητα (ρ)	7850	kg/m^3
Μάζα (M)	32.20	kg
Κύρια Κυκλική Συχνότητα (ω)	284.82	rad/s
Κύρια Συχνότητα (f)	45.33	Hz

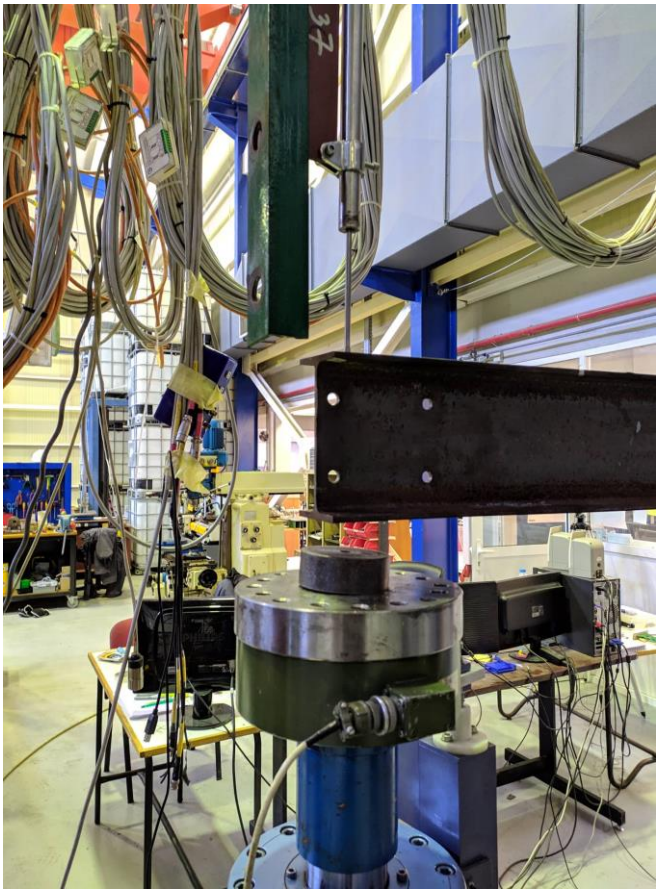
Εικόνα 4.1: Πλευρική όψη και στήριξη της δοκού για την οποία θα γίνουν οι μετρήσεις

Εικόνα 4.2: Διατομή της δοκού για την οποία θα γίνουν οι μετρήσεις

Μετά από κάποιες πρόχειρες μετρήσεις παρατηρήθηκε ότι το σφάλμα των τιμών που προκύπταν ήταν αρκετά μεγάλο, της τάξης του 10%. Η εξήγηση που δόθηκε για αυτή την παρατήρηση ήταν ότι η στήριξη της δοκούς στο άκρο της δεν ήταν πλήρης πάκτωση, όπως είχε υποτεθεί για την εφαρμογή των σχέσεων (2.1) με (2.4). Παρόλο που αυτή η ερμηνεία ήταν σωστή, δεν ήταν αρκετή ώστε να ξεκινήσουν οι μετρήσεις. Έπρεπε να υπάρχει ένα μέτρο σύγκρισης των τιμών που θα προκύπταν με μια θεωρητική τιμή. Για τον λόγο αυτό, αποφασίστηκε να υπολογιστεί πειραματικά η δυσκαμψία της δοκού.

4.3 Πειραματικός υπολογισμός δυσκαμψίας της δοκού περί τον ισχυρό άξονα

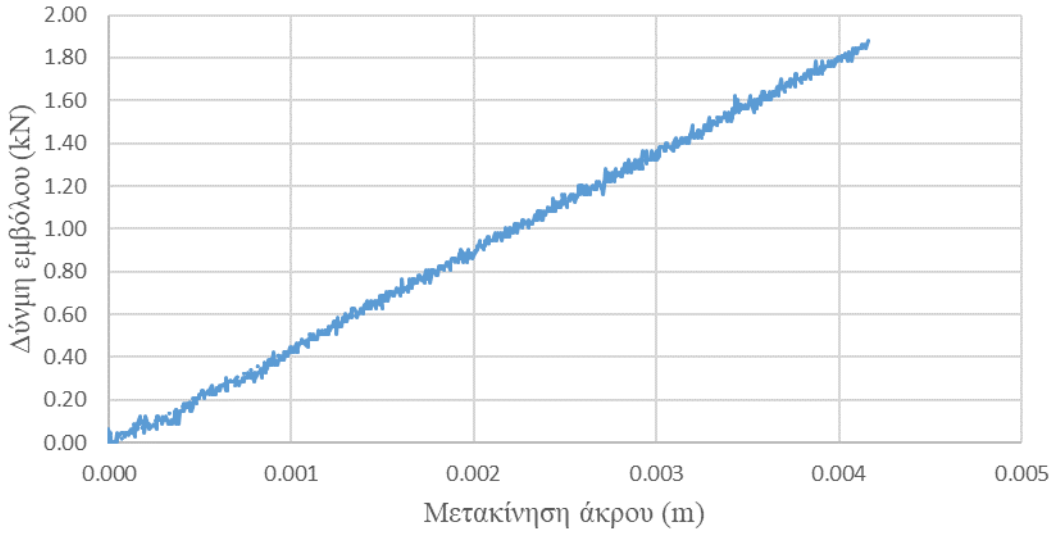
Όπως ήδη αναφέρθηκε, η ανάγκη ύπαρξης μιας θεωρητικής συχνότητας ταλάντωσης της δοκού για την αξιολόγηση της ορθότητας των μετρήσεων οδήγησε στην αναζήτηση της πραγματικής της δυσκαμψίας, αφού η προσομοίωση της στήριξής της με πλήρη πάκτωση φάνηκε να μην είναι ρεαλιστική. Για τον σκοπό αυτό τοποθετήθηκε ένα έμβολο στο άκρο της δοκού (Εικόνα 4.3). Αναφέρεται πως για την πραγματοποίηση του πειράματος αυτού έπρεπε να αλλάξει θέση η δοκός (Εικόνα 4.4). Αυτό σημαίνει ότι η δυσκαμψία της δεν θα είναι ίδια με αυτή που είχε στη προηγούμενη θέση και για αυτό όλες οι μετρήσεις που θα ακολουθήσουν θα γίνουν για τη θέση που έγινε το πείραμα



Εικόνα 4.3: Μέτρηση δυσκαμψίας μονοπροέχουσας δοκού με έμβολο

Εικόνα 4.4: Νέα θέση εξεταζόμενης δοκού

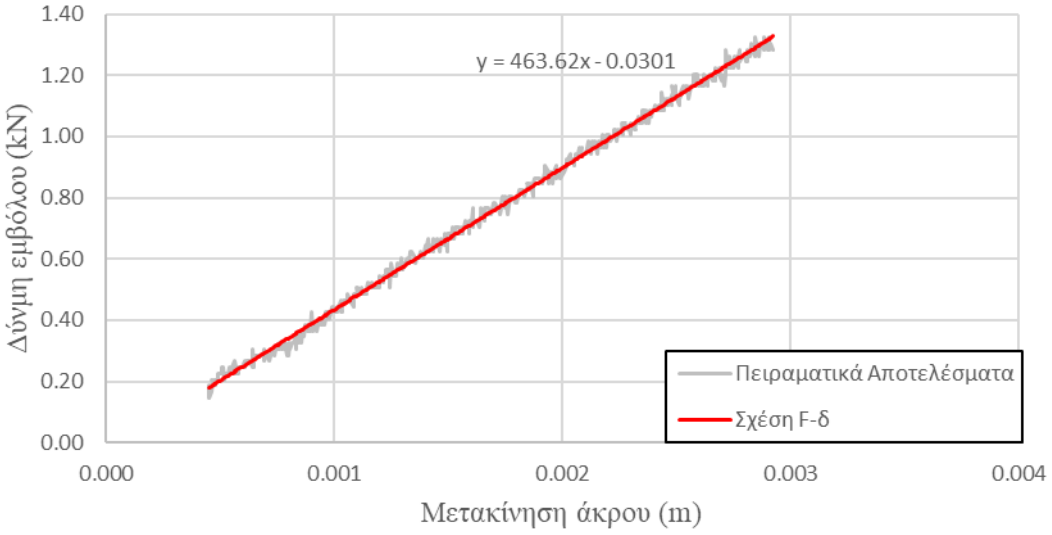
Αυξάνοντας σταδιακά της δύναμη που επέβαλε το έμβολο και μετρώντας την κατακόρυφη μετακίνηση της δοκού στην αντίστοιχη θέση μέσω ενός βελόμετρου προέκυψε ένα διάγραμμα δύναμης-μετακίνησης, η κλίση του οποίου αντιπροσωπεύει την ζητούμενη δυσκαμψία. Στο Σχήμα 4.1 παρουσιάζεται το διάγραμμα της μετακίνησης τους άκρου συναρτήσει της δύναμης του εμβόλου όπως αυτό προέκυψε από τις μετρήσεις του Εργαστηρίου.



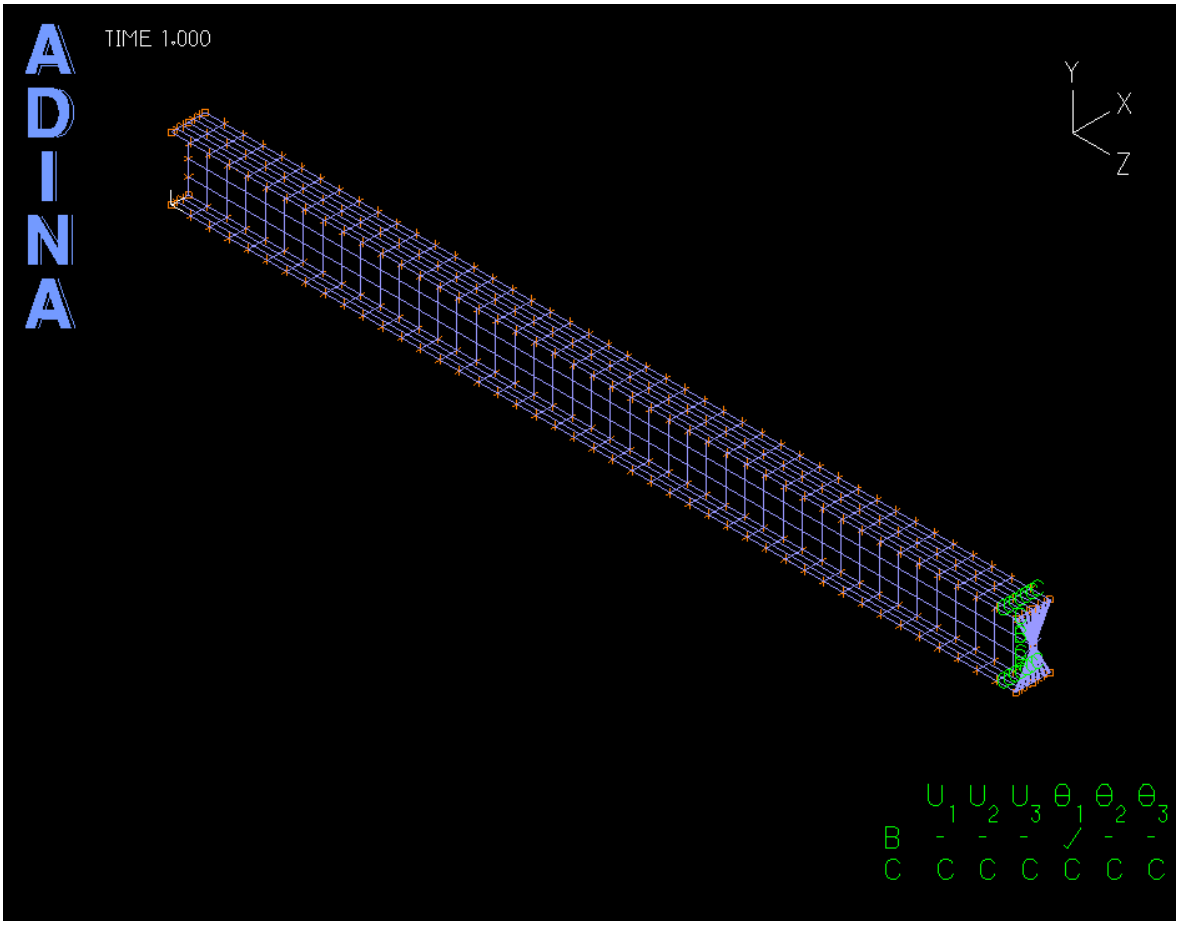
Σχήμα 4.1: Καμπύλη δύναμης-μετακίνησης όπως προέκυψε από τη πραγματοποίηση του πειράματος

Ο υπολογισμό της δυσκαμψίας αποφασίστηκε να γίνει μέσω μιας επαναληπτικής διαδικασίας δοκιμών. Με τη μέθοδο των ελαχίστων τετραγώνων χαράσσεται και υπολογίζεται η εξίσωση της ευθείας που διέρχεται από τα σημεία του γραφήματος (Σχήμα 4.2). Λόγω της αστάθειας των μετρήσεων που παρατηρείται στην αρχή και το πέρας του πειράματος εξαιρέθηκαν οι 150 πρώτες και τελευταίες τιμές του γραφήματος για τον υπολογισμό της παραπάνω εξίσωσης. Στη συνέχεια για το τυχαίο φορτίο των 2 kN υπολογίζεται η μετακίνηση του φορέα, όπως αυτή προκύπτει από τη πειραματική σχέση της δύναμης συναρτήσει της μετακίνησης. Ακολούθησε η προσομοίωση του φορέα με ένα πρόγραμμα πεπερασμένων στοιχείων. Για τον σκοπό αυτό, χρησιμοποιήθηκε το λογισμικό ADINA. Ο φορέας προσομοιώθηκε με επιφανειακά πεπερασμένα στοιχεία (Εικόνα 4.5). Το μέτρο ελαστικότητας του χάλυβα λήφθηκε 210 GPa και η πυκνότητά του 7750 kg/m^3. Το μήκος του φορέα ορίστηκε στα 2040 mm και η διατομή του είχε της διαστάσεις της πρότυπης διατομής IPE 160. Για την προσομοίωση της στήριξης, θεωρήθηκε ότι όλοι οι μεταφορικοί βαθμοί ελευθερίας είναι δεσμευμένοι, ενώ δεσμευμένες θεωρήθηκαν και οι στροφές περί τον ασθενή άξονα και περί τον ουδέτερο άξονα του φορέα. Η στροφή περί τον ισχυρό άξονα θεωρήθηκε ως άρθρωση με ένα στροφικό ελατήριο άγνωστης μέχρι στιγμής δυσκαμψίας. Η δυσκαμψία του ελατηρίου αυτού θα προκύψει με δοκιμές ώστε για το φορτίο των 2 kN στο άκρο του φορέα η αντίστοιχη μετακίνηση να ικανοποιείται τα πειραματικά αποτελέσματα.

Για φορτίο 2 kN προκύπτει ότι ο φορέας θα πρέπει να μετακινηθεί κατακόρυφα 4,38 mm. Μέσα από επαναλήψεις θα υπολογιστεί η τιμή της σταθεράς *k* του ελατηρίου που ικανοποιεί τις δύο αυτές τιμές. Μετά από δοκιμές υπολογίστηκε ότι η ζητούμενη δυσκαμψία είναι 10900 kNm/rad. Για την δυσκαμψία αυτή, θα εκτελεστεί η ιδιομορφική ανάλυση (Frequencies/Modes στο ADINA) η οποία θα δώσει τις συχνότητες και τα σχήματα των ιδιομορφών του φορέα. Η κύρια μεταφορική ιδιομορφή περί τον ισχυρό άξονα έχει συχνότητα 39.16 Hz και το σχήμα της φαίνεται στην Εικόνα 4.7.



Σχήμα 4.2: Υπολογισμός συνάρτησης Δύναμης-Μετακίνησης με βάση τα αποτελέσματα του πειράματος

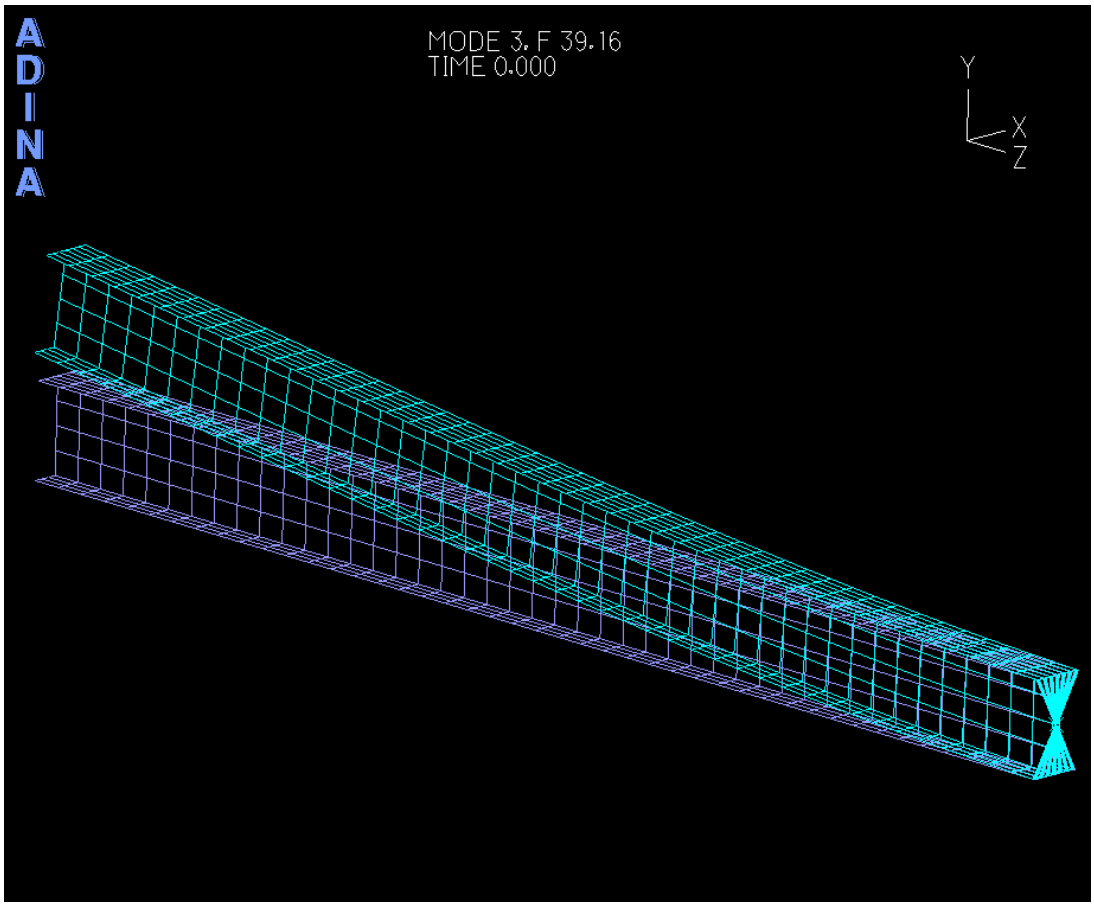


Εικόνα 4.5: Προσομοίωση της δοκού με το λογισμικό ADINA

```
ADINA: AUI version 9.4.2, 3 July 2019:  *** NO HEADING DEFINED ***
Licensed from ADINA R&D, Inc.
Finite element program ADINA, response range type load-step:
Scanning for absolute maximum in zone WHOLE_MODEL:
Variable Y-DISPLACEMENT:

-4.35900E-03, time 1.00000E+00, node 225

*** End of list.
```

Εικόνα 4.6: Μέγιστη μετακίνηση του φορέα περί τον ισχυρό του άξονα (Y στο μοντέλο) όπως προκύπτει για ελατήριο σταθεράς k=10900 kNm/rad.

Όπως αναμενόταν, η συχνότητα ταλάντωσης που προέκυψε δεν είχε καμία σχέση με αυτή που υπολογίστηκε από τη θεώρηση ότι το δεσμευμένο άκρο της δοκού είναι πλήρως πακτωμένο. Στη συνέχεια, ακολουθεί μια σειρά μετρήσεων όπου συγκρίνονται τα αποτελέσματα μεταξύ τους αλλά και με την τιμή που υπολογίστηκε στο παρόν κεφάλαιο. Επιπλέον, θα εξεταστούν τα αποτελέσματα και με τις 3 εκδοχές του αλγορίθμου που αναπτύχθηκε, ώστε να αξιολογηθεί η αποτελεσματικότητά τους.



Εικόνα 4.7: Πρώτη μεταφορική ιδιομορφή περί τον ισχυρό άξονα (Y) όπως προέκυψε από το λογισμικό ADINA

4.4 Πρώτη σειρά μετρήσεων

Κατά την πρώτη προσπάθεια εντοπισμού της συχνότητας ταλάντωσης του προβόλου χρησιμοποιήθηκε η βασική εκδοχή του αλγορίθμου που αναπτύχθηκε. Η εκδοχή αυτή, περιλαμβάνει παρουσίαση των επιταχύνσεων σε πραγματικό χρόνο και γρήγορο μετασχηματισμό Fourier των τιμών τους ως έχουν. Δεδομένου πως η αναμενόμενες συχνότητες θα είναι αρκετά μεγαλύτερες από αυτές που μετρήθηκαν στην προηγούμενη εφαρμογή με τον χάρακα, υπήρχε ανάγκη για σχετικά μεγάλες συχνότητες καταγραφής. Αύξηση της συχνότητας καταγραφής ισοδυναμεί με μεγαλύτερη ακρίβεια των αποτελεσμάτων. Για την επίτευξη μιας σχετικά μεγάλης ταχύτητας καταγραφής, αυξήθηκε η ταχύτητα τη σειριακής επικοινωνίας μεταξύ Teensy και υπολογιστή στα 500000 baud rate. Η συχνότητα με την οποία έγιναν οι καταγραφές του παρόντος κεφαλαίου είναι τα 204 Hz. Τα αποτελέσματα των καταγραφών παρουσιάζονται στον Πίνακα 4.2.

Πίνακας 4.2: Αποτελέσματα πρώτης σειράς μετρήσεων για ταλαντώσεις περί τον ισχυρό άξονα

α/α	Συχνότητα περί τον ισχυρό άξονα (Hz)	Απόκλιση Συχνότητας (%)
1	37.76	4.34
2	35.78	1.13
3	35.59	1.66
4	36.06	0.36
5	38.11	5.31
6	35.4	2.18
7	36.72	1.46
8	37.55	3.76
9	37.39	3.32
10	34.54	4.56
11	34.65	4.26
12	34.67	4.20
13	37.88	4.67
14	34.41	4.92
15	36.67	1.33

Για τα παραπάνω αποτελέσματα, χρησιμοποιήθηκε μια σειρά διαφορετικών διεγέρσεων ώστε να εξεταστεί η αποτελεσματικότητα του αλγορίθμου στην εύρεση της ζητούμενης συχνότητας ταλάντωσης. Οι διεγέρσεις αυτές ήταν:

1. Χτύπημα της δοκού με το χέρι στο μέσο ή στο άκρο της
2. Κύλιση της γερανογέφυρας του εργαστηρίου
3. Χτύπημα της δοκού με σφυρί στο άκρο της

Από τα παραπάνω αποτελέσματα παρατηρείται ότι υπήρχαν περιπτώσεις που το σφάλμα σε σχέση με τη θεωρητική τιμή της συχνότητας ήταν μικρό (0.36%). Παρόλα αυτά, θα πρέπει να αναφερθεί πως υπάρχει μεγάλη διασπορά των τιμών, γεγονός που δεν εμπνέει εμπιστοσύνη για την αξιοπιστία των αποτελεσμάτων. Το συμπέρασμα αυτό ενισχύεται από το ότι υπάρχουν μετρήσεις στις οποίες το σφάλμα πλησίασε το 5 %. Επιπλέον, υπολογίστηκε η μέση τιμή στα 36,21 Hz και η τυπική απόκλιση των παραπάνω τιμών στα 1,32 Hz. Οι τιμές αυτές θα είναι χρήσιμες για τη σύγκριση των αποτελεσμάτων με αυτά που θα προκύψουν σε μετέπειτα μετρήσεις.

4.5 Δεύτερη σειρά μετρήσεων

Παρόλο που οι παραπάνω μετρήσεις έγιναν με σταθερή συχνότητα καταγραφής τα 204 Hz, δεν υπήρχε μια δικλείδα ασφαλείας που να εγγυάται ότι οι μετρήσεις έχουν το σταθερό χρονικό βήμα που απαιτεί ο μετασχηματισμός Fourier. Για τον λόγο αυτό, οι μετρήσεις που θα ακολουθήσουν θα γίνουν με την δεύτερη εκδοχή τους αλγορίθμου που αναπτύχθηκε στη γλώσσα Python. Κατά την εκδοχή αυτή, γίνεται μια γραμμική παρεμβολή μεταξύ όλων των καταγεγραμμένων σημείων ώστε να υπάρχει γνώση του χρονικού βήματος των τιμών με τις οποίες γίνεται ο μετασχηματισμός Fourier. Επιπλέον, η παρουσίαση ων τιμών των επιταχύνσεων δεν γίνεται σε πραγματικό χρόνο ώστε να γίνει εξοικονόμηση της μνήμης RAM του υπολογιστή και επίτευξη μεγαλύτερων συχνοτήτων

καταγραφής. Με δεδομένα αυτά, οι καταγραφές που θα παρουσιαστούν στον Πίνακα 4.3, έγιναν με συχνότητα καταγραφής 225 Hz.

Πίνακας 4.3: Αποτελέσματα δεύτερης σειράς μετρήσεων για ταλαντώσεις περί τον ισχυρό άξονα

α/α	Συχνότητα περί τον ισχυρό άξονα (Hz)	Απόκλιση Συχνότητας (%)
1	39.16	0.00
2	39.27	0.28
3	39.32	0.41
4	39.23	0.18
5	39.21	0.13
6	39.35	0.49
7	39.28	0.31
8	39.37	0.54
9	39.34	0.46
10	39.33	0.43
11	39.29	0.33
12	39.13	0.08
13	39.31	0.38
14	39.21	0.13
15	39.21	0.13

Για τις παραπάνω καταγραφές υπολογίστηκε η μέση τιμή στα 39,27 Hz και η τυπική απόκλιση στα 0,07 Hz. Παρατηρείται πως για τις δεδομένες μετρήσεις το σφάλμα οριακά ξεπέρασε το 0,5 %, ενώ υπήρχε και περίπτωση με μηδενικό σφάλμα. Θα πρέπει να αναφερθεί πως κάτι τέτοιο είναι καθαρά τυχηματικό καθώς υπάρχουν αρκετές αβεβαιότητες που διέπουν τον φορέα, όπως το πραγματικό μέτρο ελαστικότητας του χάλυβα, η πραγματική πυκνότητα του και οι πραγματική γεωμετρία. Επιπροσθέτως, οι τιμές έχουν πολύ μικρή απόκλιση μεταξύ τους σε σχέση με τα αποτελέσματα των προηγούμενων μετρήσεων. Αυτή η μικρή απόκλιση των τιμών δείχνει τη σταθερή λειτουργία του συγκεκριμένου αλγορίθμου κάτω από διαφορετικές συνθήκες διέγερσης.

4.6 Τρίτη σειρά μετρήσεων

Ακολουθεί η τρίτη εκδοχή του αλγορίθμου που αναπτύχθηκε. Οι καταγραφές θα αποθηκευτούν από το πρόγραμμα Arduino σε ένα αρχείο txt και η επεξεργασία τους θα γίνει ξεχωριστά μέσω ενός κώδικα σε γλώσσα Python. Με τον τρόπο αυτό επιτυγχάνονται υψηλές ταχύτητες καταγραφής, αλλά χάνεται ο αυτοματισμός και η δυνατότητα παρουσίασης των αποτελεσμάτων σε πραγματικό χρόνο. Όλα αυτά οδήγησαν σε μια συχνότητα καταγραφής 1000 Hz, αρκετά υψηλή ώστε οι ταλαντώσεις να καταγράφονται με αξιοσημείωτη ακρίβεια. Τα αποτελέσματα αυτών των μετρήσεων παρουσιάζονται στον Πίνακα 4.4.

Πίνακας 4.4: Αποτελέσματα τρίτης σειράς μετρήσεων για ταλαντώσεις περί τον ισχυρό άξονα

α/α	Συχνότητα περί τον ισχυρό άξονα (Hz)	Απόκλιση Συχνότητας (%)
1	38.97	0.49
2	39.37	0.54
3	39.29	0.33
4	38.97	0.49
5	38.72	1.12
6	39.13	0.08
7	39.29	0.33
8	39.29	0.33
9	39.21	0.13
10	39.37	0.54
11	39.21	0.13
12	39.21	0.13
13	39.29	0.33
14	39.29	0.33
15	39.13	0.08

Από τις παραπάνω τιμές προκύπτει μια μέση τιμή στα 39,18 Hz και τυπική απόκλιση στα 0,18 Hz. Από τα αποτελέσματα της συγκεκριμένης σειράς μετρήσεων παρατηρείται ότι δεν υπήρχαν αισθητές διαφορές με αυτά της δεύτερης σειράς. Μπορεί η μέση τιμή των καταγραφών να ήταν πλησιέστερα στη θεωρητική αλλά υπήρχε μια σχετικά μεγαλύτερη απόκλιση των τιμών. Φαίνεται, δηλαδή, ότι η ταχύτερη συχνότητα καταγραφής δεν εξασφάλισε αισθητά καλύτερα αποτελέσματα μιας και ήδη από τα 225 Hz οι μετρήσεις είχανε σταθεροποιηθεί και τα αποτελέσματα πλησίασαν τη θεωρητική συχνότητα ταλάντωσης.

4.7 Εύρεση συντελεστή απόσβεσης ταλάντωσης

Στην προσπάθεια υπολογισμού του συντελεστή απόσβεσης της ταλάντωσης της δοκού περί τον ισχυρό της άξονα χρησιμοποιήθηκαν δέκα καταγραφές από την τρίτη σειρά μετρήσεων καθώς ήταν πιο ακριβείς ως προς τη συχνότητα καταγραφής τους σε σχέση με τις άλλες δυο. Από κάθε καταγραφή προέκυψαν δύο συντελεστές απόσβεσης, σύμφωνα με τον αλγόριθμο που αναπτύχθηκε για τον εντοπισμό τους, ενώ υπολογίστηκε και η μέση τιμή αυτών. Τα αποτελέσματα αυτών των μετρήσεων παρουσιάζονται στον Πίνακα 4.5.

Από τα παρακάτω αποτελέσματα (Πίνακας 4.5) προκύπτει μέση τιμή του συντελεστή απόσβεσης 0,67 % και τυπική απόκλιση 0,84 %. Παρατηρείται ότι η απόκλιση των τιμών είναι αρκετά μεγάλη και αυτό οφείλεται στις τιμές από τις καταγραφές 2, 8, 9. Αν εξαιρεθούν οι μετρήσεις αυτές η μέση τιμή του συντελεστή απόσβεσης είναι 0,25 % και η μέση τιμή του 0,02 %, αρκετά μικρότερη σε σχέση με το 0,84 %. Στην προσπάθεια ερμηνείας της απόκλισης από τη μέση τιμή που προέκυψε για κάποιες καταγραφές, παρουσιάζονται στον Πίνακα 4.6 τα εύρη των τιμών των επιταχύνσεων μέσα στα οποία κινήθηκαν οι ταλαντώσεις.

Από τον Πίνακας 4.6 εντοπίζεται πως οι μεγάλες τιμές των συντελεστών απόσβεσης παρατηρήθηκαν σε ταλαντώσεις με μικρό εύρος επιταχύνσεων. Αυτό, βέβαια, δεν αποτελεί κανόνα καθώς υπήρχαν διαταραχές μικρότερου εύρους με μικρότερη απόσβεση. Από αυτό προκύπτει ότι ο τρόπος διαταραχής του φορέα από τη θέση ισορροπίας του και το μέγεθος των μετακινήσεων που προκαλεί μπορεί να επηρεάσει σημαντικά τα

αποτελέσματα των μετρήσεων, καθώς οι διαταραχές με μικρό εύρος επιταχύνσεων αντιστοιχούν στη διαταραχή του φορέα μέσω του χτυπήματος του με ένα σφυρί.

Πίνακας 4.5: Εύρεση συντελεστή απόσβεσης για ταλαντώσεις περί τον ισχυρό άξονα

α/α	1η τιμή απόσβεσης (%)	2η τιμή απόσβεσης (%)	Μέση τιμή απόσβεσης (%)
1	0.26	0.26	0.26
2	0.76	0.65	0.71
3	0.25	0.27	0.26
4	0.12	0.3	0.21
5	0.26	0.26	0.26
6	0.25	0.28	0.27
7	0.26	0.29	0.28
8	1.53	1.2	1.37
9	2.89	2.78	2.84
10	0.26	0.24	0.25

Πίνακας 4.6: Εύρη επιταχύνσεων συναρτήσει των υπολογισμένων συντελεστών αποσβέσεως

α/α	Μέγιστη επιτάχυνση (g)	Εύρος τιμών επιταχύνσεων (g)	Μέση τιμή απόσβεσης (%)
1	2	0.96	0.26
2	1.24	0.2	0.71
3	2	0.96	0.26
4	2	0.97	0.21
5	2	0.96	0.26
6	1.28	0.24	0.27
7	1.22	0.18	0.28
8	1.11	0.07	1.37
9	1.35	0.31	2.84
10	2	0.96	0.25

4.8 Εύρεση κύριας συχνότητας ταλάντωσης υπό μικρότερες δυνάμεις προεντάσεως κοχλία στην στήριξη

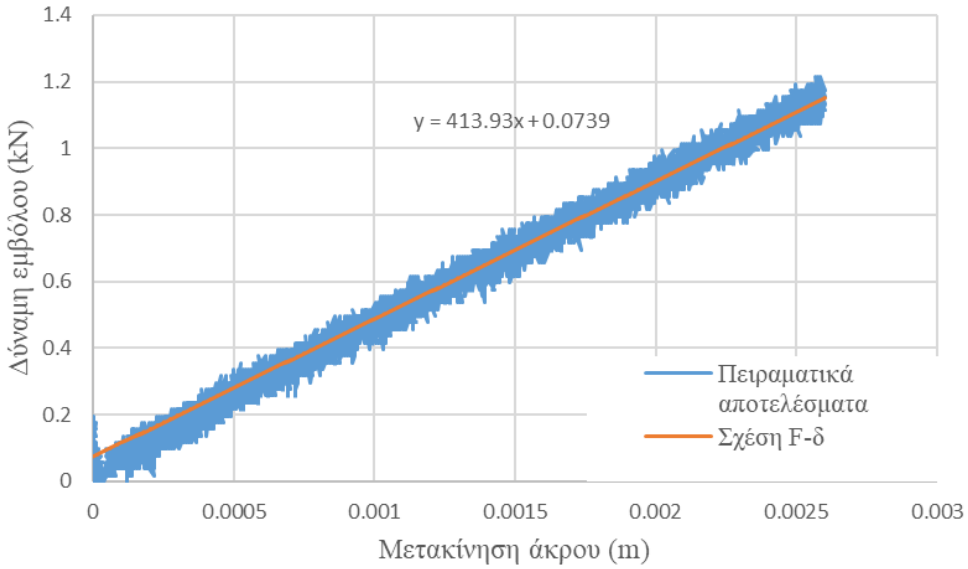
Σε μια επιπλέον προσπάθεια ελέγχου της αξιοπιστίας του αισθητήρα και των αλγορίθμων πίσω από αυτόν αποφασίστηκε να γίνει μια νέα σειρά μετρήσεων. Στόχος ήταν η ανίχνευση των συχνοτήτων ταλάντωσης του προβόλου μεταβάλλοντας σταδιακά την δύναμη προέντασης του ενός από του τέσσερις κοχλίες που τον στηρίζουν (Εικόνα 4.8). Τα αποτελέσματα μιας τέτοιας σειράς δοκιμών μπορεί να αποτελέσει γνώμονα για το επίπεδο ευαισθησίας του αισθητήρα. Δεδομένου ότι αισθητήρες σαν αυτόν της παρούσας εργασίας χρησιμοποιούνται για την ανίχνευση βλαβών μετά από κάποια φυσική καταστροφή ή για λόγους συντήρησης, κρίθηκε απαραίτητο να γίνει η συγκεκριμένη σειρά δοκιμών. Το χαλάρωμα του κοχλία αντιπροσωπεύει μια βλάβη που θα μπορούσε να συμβεί σε βάθους χρόνους στον πρόβολο.

Εικόνα 4.8: Χαλάρωση προέντασης του πάνω δεξιά κοχλία

Για τη δεδομένη σειρά μετρήσεων έγιναν δοκιμές για τρεις διαφορετικές προεντάσεις του κοχλία. Ο έλεγχος των επιβαλλόμενων προεντάσεων έγινε μέσω ενός ροπόκλειδου, όπου η πρώτη έγινε για την αρχική προένταση με ροπή 300 Nm, η δεύτερη για 100 Nm και η τρίτη για την ελάχιστη δυνατή που μπορούσε να επιβληθεί, δηλαδή 60 Nm. Πριν τις μετρήσεις έγινε και πειραματικός υπολογισμός της δυσκαμψίας της σύνδεσης. Υπολογίζοντας την δυσκαμψία του προβόλου για κάθε περίπτωση χωριστά υπήρχε η δυνατότητα να γίνει σύγκριση των αποτελεσμάτων με αυτά που προκύπτουν από ένα πρόγραμμα πεπερασμένων στοιχείων. Ο υπολογισμός της δυσκαμψίας έγινε μέσω ενός εμβόλου που επέβαλλε σταδιακά στο άκρο της δοκού μια συγκεντρωμένη δύναμη και δημιουργούσε μια μετακίνηση η οποία μετρούταν ακριβώς στο ίδιο σημείο. Θα πρέπει να αναφερθεί πως για κάθε προένταση έγιναν τρεις τέτοιες φορτίσεις και αποφορτίσεις για λόγους ακρίβειας. Ομοίως με τις προηγούμενες μετρήσεις προέκυψε για κάθε προένταση ένα διάγραμμα δύναμης μετακίνησης όπου μετά από μια σειρά δοκιμών στο λογισμικό ADINA υπολογίστηκε η αντίστοιχη δυσκαμψία και στη συνέχεια η συχνότητα ταλάντωσης.

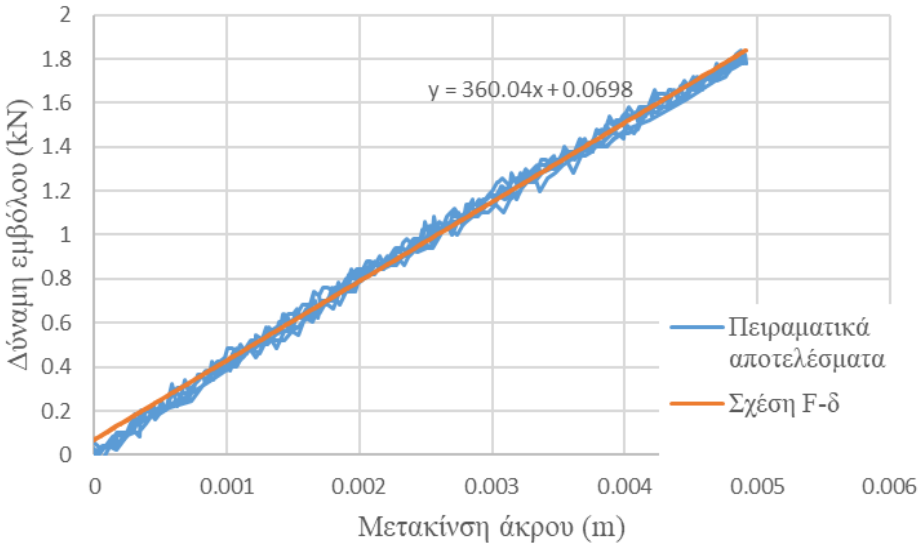
Για την ροπή προέντασης των 300 Nm έχει ήδη υπολογιστεί η δυσκαμψία του φορέα και η κύρια συχνότητα ταλάντωσής του περί τον ισχυρό άξονα παρόλα αυτά επαναλήφθηκε το πείραμα και οι μετρήσεις καθώς μεσολάβησε σημαντικό χρονικό διάστημα από τη πρώτη σειρά μετρήσεων. Με τη μέθοδο των ελαχίστων τετραγώνων χαράσσεται η κοινή ευθεία διαγράμματος δύναμης-μετακίνησης και υπολογίζεται η μετακίνηση που προκαλεί η επιβολή ενός φορτίου 2 kN (Σχήμα 4.3). Για το φορτίο αυτό η μετακίνηση που του αντιστοιχεί είναι 4,64 mm, ενώ η δυσκαμψία που πρέπει να έχει ο φορέας για το ζεύγος αυτό είναι 8600 kNm/rad. Από τα δεδομένα αυτά προέκυψε κύρια συχνότητα ταλάντωσης του φορέα ίση με 37,93 Hz. Μετά την εκτέλεση του πειράματος έγιναν τρεις μετρήσεις από τις οποίες υπολογίστηκε η συχνότητα ταλάντωσης που δίνει ο

αισθητήρας περί τον ισχυρό άξονα (άξονας Z του επιταχυνσιομέτρου) καθώς και η απόκλιση των τιμών αυτών από αυτή που έδωσε το λογισμικό ADINA.

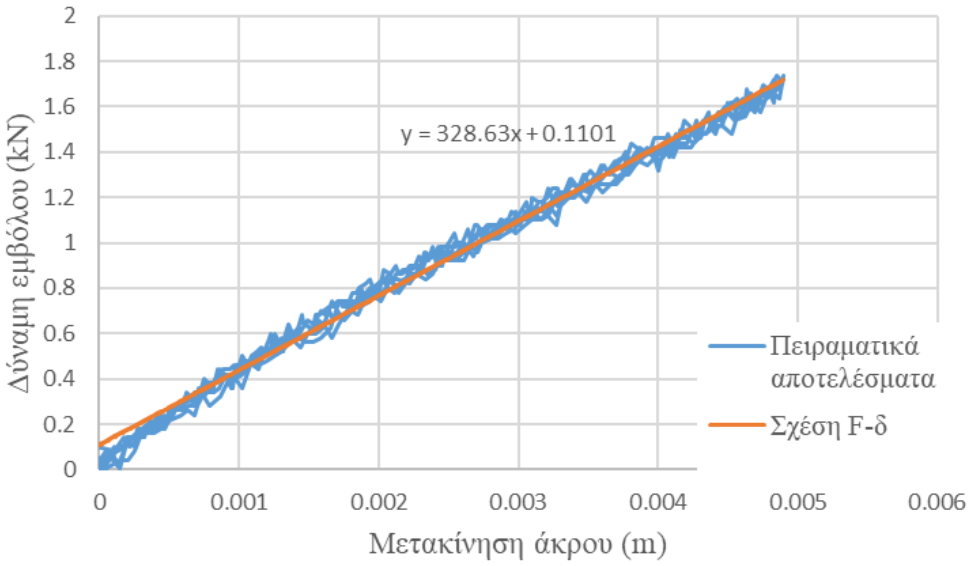


Σχήμα 4.3: Υπολογισμός σχέσης Δύναμης-Μετακίνησης με βάση τα αποτελέσματα του πειράματος για προένταση κοχλία με ροπή 300 Nm

Η ίδια διαδικασία ακολουθήθηκε και για τις άλλες δύο τιμές προέντασης του πάνω δεξιά κοχλία. Για την περίπτωση της προέντασης με 100 Nm για φορτίο 2 kN αντιστοιχεί μετακίνηση 5,36 mm και δυσκαμψία 4900 kNm/rad (Σχήμα 4.4). Τέλος, για προένταση με 60 Nm υπολογίστηκε ότι η μετακίνηση για φορτίο 2 kN είναι 5,75 mm και η δυσκαμψία 4000 kNm/rad (Σχήμα 4.5). Στον πίνακα που ακολουθεί (Πίνακας 4.7) παρουσιάζονται τα αποτελέσματα των τριών μετρήσεων για κάθε τιμή προέντασης καθώς και οι αντίστοιχες αποκλίσεις από τις τιμές του λογισμικού ADINA.



Σχήμα 4.4: Υπολογισμός σχέσης Δύναμης-Μετακίνησης με βάση τα αποτελέσματα του πειράματος για προένταση κοχλία με ροπή 100 Nm



Σχήμα 4.5: Υπολογισμός σχέσης Δύναμης-Μετακίνησης με βάση τα αποτελέσματα του πειράματος για προένταση κοχλία με ροπή 60 Nm

Πίνακας 4.7: Εύρεση συντελεστή απόσβεσης για ταλαντώσεις περί τον ισχυρό άξονα

Ροπή προεντάσεως (Nm)	Ιδιοσυχνότητα λογισμικού ADINA (Hz)	Ιδιοσυχνότητα αισθητήρα (Hz)	Απόκλιση (%)
300	37.93	38.51	1.53
		38.41	1.27
		38.68	1.98
100	34.8	38.42	10.40
		38.60	10.92
		38.49	10.60
60	33.4	38.69	15.84
		38.67	15.78
		38.83	16.26

Παρατηρώντας τις παραπάνω τιμές, είναι εμφανές πως ο αισθητήρας δεν κατάφερε ανιχνεύσει την αλλαγή που συνέβη στη σύνδεση της δοκού. Ακόμα για προένταση κοχλία στα 100 Nm η απόκλιση της τάξης του 10 % θεωρείται απαγορευτική για να θεωρηθεί πως λειτουργεί με ακρίβεια το επιταχυνσιόμετρο.

4.9 Συμπεράσματα

Από τις τρεις σειρές μετρήσεων που πραγματοποιήθηκαν συμπεραίνεται ότι απαραίτητη προϋπόθεση για να είναι τα αποτελέσματα των καταγραφών σταθερά είναι να υπάρχει μια σταθερή συχνότητα καταγραφής και ένα όσο το δυνατόν σταθερό χρονικό βήμα. Ακόμα και αν είναι σταθερό το πλήθος των τιμών που λαμβάνονται σε ένα δεδομένο χρονικό διάστημα υπάρχει πιθανότητα να μην ισαπέχουν χρονικά όλες αυτές οι τιμές μεταξύ τους και άρα να μην λειτουργεί σωστά ο μετασχηματισμός Fourier. Η

παρεμβολή τιμών στις ήδη υπάρχουσες μετρήσεις είναι μια λύση σε αυτό το πρόβλημα με το μειονέκτημα ότι εισάγεται ένα σφάλμα στα αποτελέσματα. Οι δοκιμές που έγιναν έδειξαν ότι το σφάλμα αυτό δεν είναι μεγάλο συγκριτικά με το πως βελτιώνονται τα αποτελέσματα. Σε ότι αφορά την εύρεση του συντελεστή απόσβεσης, υπήρχαν περιπτώσεις που οι τιμές απέκλιναν από τη μέση τιμή που υπολογίστηκε λόγο της φύσης της επιβαλλόμενης διαταραχής. Επισημαίνεται ότι μπορεί η επίτευξη υψηλών συχνοτήτων καταγραφής να μην έδειξε σημαντικά αποτελέσματα στη συγκεκριμένη εφαρμογή, αλλά μπορεί να αποτελέσει χρήσιμο εργαλείο για τη καταγραφή ταλαντώσεων σε πιο δύσκαμπτους φορείς ή για την προσπάθεια ανίχνευσης υψηλότερων ιδιοσυχνοτήτων πέρα της κύριας. Τέλος, η δοκιμή μου έγινε για την αξιολόγηση της ικανότητας του αισθητήρα να εντοπίζει αλλαγές στο φορέα έδωσε σημαντικά αποτελέσματα. Η χαλάρωση της προέντασης του ενός από τους τέσσερις κοχλίες έδειξε ότι επηρεάζει την συμπεριφορά του μέλους. Παρόλα αυτά οι αισθητήρας της παρούσας εργασίας δεν κατάφερε να ανιχνεύσει την αλλαγή αυτή, λόγω της φύσης του προβλήματος. Οι μικρές μετακινήσεις που επιβλήθηκαν στον φορέα σε συνδυασμό με την παρουσία τριβών στη σύνδεση δεν επιτρέπουν τον εντοπισμό της όποιας μείωσης της δυσκαμψίας επιβλήθηκε.

5 Συμπεράσματα

Ο στόχος της παρούσας διπλωματικής εργασίας συνίσταται στην ανάπτυξη ενός ολοκληρωμένου συστήματος που θα μπορεί να καταγράφει ταλαντώσεις και ταυτόχρονα να επεξεργάζεται τις τιμές των αναπτυσσόμενων επιταχύνσεων παρέχοντας πληροφορίες για τα χαρακτηριστικά των φορέων. Πέρα από αυτό, λήφθηκε σημαντικά υπόψη και η παράμετρος του κόστους των εξαρτημάτων που θα αποτελούσαν το σύστημα αυτό. Δεδομένου ότι η εφαρμογή αισθητήρων και η παρακολούθηση των μετρήσεών τους στις κατασκευές είναι αρκετά δαπανηρή, ήταν σημαντικό τα εξαγόμενα αποτελέσματα να εμπνέουν αξιοπιστία.

Κατά τον προγραμματισμό του συστήματος αυτού παρατηρήθηκε η ανάγκη ύπαρξης μεγάλης υπολογιστική ισχύος. Στην προσπάθεια για ταυτόχρονη συλλογή και επεξεργασία των δεδομένων, η απλότητα του κώδικα και η σωστή οργάνωσή του ήταν καθοριστική για την επίτευξη μεγάλων συχνοτήτων λειτουργίας. Επιπλέον, η αυτοματοποίηση της λειτουργίας του κώδικα εισήγαγε επιπλέον υπολογιστικό φόρτο. Αυτό είχε ως αποτέλεσμα να πρέπει να γίνει επιλογή ανάμεσα στις υψηλές συχνότητες καταγραφής των δεδομένων και την διευκόλυνση του χρήστη για την εκτέλεση μιας μέτρησης. Οι υψηλές συχνότητες καταγραφής ισοδυναμούν με ακρίβεια των αποτελεσμάτων, παρόλα αυτά παρατηρήθηκε πως από κάποια συχνότητα και μετά δεν υπήρχε μείωση στις αποκλίσεις των μετρήσεων.

Σε ότι αφορά τις μετρήσεις που έγιναν πάνω στο μεταλλικό έλασμα παρατηρήθηκε πως η σταθερή συχνότητα καταγραφής ήταν το κλειδί για να προκύψει ένας αλγόριθμος με αξιόπιστα αποτελέσματα. Επιπλέον, φάνηκε πως το σύστημα που αναπτύχθηκε συμπεριφέρεται ικανοποιητικά για μεγάλο εύρος δυσκαμψιών. Έχοντας λάβει υπόψη τα αποτελέσματα του πρώτου μοντέλου, ο κώδικας βελτιώθηκε ώστε να επιτευχθούν και σταθερότερες αλλά και υψηλότερες συχνότητες καταγραφής. Με τη συχνότητα καταγραφής να ξεπερνά τα 1000 Hz, ο αλγόριθμος έδωσε πολύ μικρά σφάλματα στον υπολογισμό της συχνότητας ταλάντωσης και του συντελεστή απόσβεσης της μονοπροέχουσας δοκού. Αντιθέτως, τα αποτελέσματα του συστήματος δεν φάνηκε να επηρεάζονται από τις όποιες αλλαγές επιβλήθηκαν στον φορέα. Αυτό οφείλεται στο γεγονός πως οι αλλαγές που εισήχθησαν στον φορέα δεν ήταν αρκετά σημαντικές ώστε να γίνουν αισθητές σε μικρά εύρη μετακινήσεων.

Λαμβάνοντας υπόψη ότι το κόστος των εξαρτημάτων του συστήματος δεν ξεπέρασε τα 50 €, είναι πολύ αισιόδοξο το γεγονός ότι στις πιο πολλές μετρήσεις δεν σημειώθηκαν μεγάλα σφάλματα και τα αποτελέσματα ήταν τα αναμενόμενα. Παρόλα αυτά, θα ήταν αναγκαίο να γίνουν δοκιμές σε ένα μεγαλύτερο πλήθος φορέων ώστε να αξιολογηθούν αποτελέσματα από διάφορα στατικά συστήματα και υλικά. Κάτι τέτοιο, βέβαια, θα απαιτούσε τη χρήση ποιοτικότερων υλικών για την συνδεσμολογία των εξαρτημάτων, αυξάνοντας έτσι το κόστος. Επιπλέον, η χρήση ενός επιταχυνσιομέτρου μεγαλύτερης ακρίβειας ίσως μείωνε και άλλο τις όποιες αποκλίσεις από θεωρητικές τιμές προέκυψαν. Η αγορά είναι γεμάτη από διαθέσιμες επιλογές, αυτό που θα πρέπει να λάβει κανείς υπόψη είναι τα επίπεδα σφάλματος που επιθυμεί στις μετρήσεις του και τα χρήματα που μπορεί να διαθέσει.

6 Βιβλιογραφία

1. Δημητρακόπουλος Θ. (2016). «Τα πρωτόκολλα επικοινωνίας I2C και SPI και η υλοποίησή τους σε σύστημα τηλεχειρισμού, με τη χρήση της ηλεκτρονικής πλατφόρμας προτυποποίησης Arduino.», Μεταπτυχιακή Διατριβή, Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
2. Εμμανουήλ Γ., Αγγελόπουλος Δ. (2012), «Πρωτόκολλα διασύνδεσης αισθητήρων με υπολογιστή», Πτυχιακή Διατριβή, Τεχνολογικό Εκπαιδευτικό Ίδρυμα Λαμίας-Τμήμα Ηλεκτρονικής
3. Σταυρακάκης Μ. Ν. (2016). «Μερικές Διαφορικές Εξισώσεις & Μιγαδικές Συναρτήσεις», Αυτοέκδοση, Αθήνα
4. Chopra A.K. (2008). «Δυναμική των Κατασκευών» (3η Έκδοση), Εκδόσεις Μ. Γκιούρδας
5. Rainieri C., Fabbrocino G. (2014), «Operational Modal Analysis of Civil Engineering Structures», Springer
6. Papanikolas P., Stathopoulos-Vlamis A., Panagis A. (2011), «The structural health monitoring system of Rion Antirion Bridge "Charilaos Trikoupis"», International Conference on Bridges and Soil-Bridge Interaction
7. Bergland G.D. (1969), «A guided tour of the fast Fourier transform», Institute of Electrical and Electronics Engineers
8. Varanis M., Silva A.L., Brunetto P.H., Gregolin R.F. (2015), «Instrumentation for mechanical vibrations analysis in the time domain and frequency domain using the Arduino platform», Faculty of Engineering, Federal University of Grande Dourados, Dourados, MS, Brazil
9. Monitoring the Structural Health of the Rion-Antirion Bridge Using the LabVIEW Real-Time Module, http://sine.ni.com/cs/app/doc/p/id/cs-689# [20/8/19]
10. MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Devices, https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/ [20/6/19]
11. Wikipedia (2019), Serial Communication, https://en.wikipedia.org/wiki/Serial_communication [20/6/19]
12. I2C, https://en.wikipedia.org/wiki/I²C [20/6/19]
13. The Electronics, MPU6050 interfacing with Arduino, https://theelectronics.co.in/mpu6050-interfacing-arduino/ [10/7/19]
14. Teensyduino, https://www.pjrc.com/teensy/teensyduino.html [10/7/19]
15. JetBrains, PyCharm, https://www.jetbrains.com/pycharm/ [11/7/19]
16. NTi Audio, Fast Fourier Transform FFT – Basics, https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft [13/7/19]
17. O'Connor J. J., Robertson E. F., «Joseph Fourier», http://www-history.mcs.st-andrews.ac.uk/, University of St Andrews [3/9/19]

Παράρτημα Α. Κώδικας λειτουργίας αισθητήρα

```
#include <Wire.h>

unsigned long time; // ορισμός μεταβλητής χρόνου

const int MPU_addr = 0x68; // Η εξ ορισμού διεύθυνση του αισθητήρα MPU 6050 είναι 0x68

void setup() {

    // Έναρξη επικονωνίας με τον δίαυλο I2C
    Serial.begin(500000); // Ορισμός ταχύτητας σειριακής επικοινωνίας
    Wire.begin();

    Wire.beginTransmission(MPU_addr); // Πρώτη επικοινωνία με τον αισθητήρα
    Wire.write(0x6B);
    Wire.write(0x00);
    Wire.endTransmission(true);

    Wire.beginTransmission(MPU_addr); // Ορισμός των 2 g ως μέγιστης μετρούμενη επιτάχυνσης
    Wire.write(0x1C);
    Wire.write(0x00);
    Wire.endTransmission(true);
}

void loop() {

    // Έναρξη ρουτίνας λειτουργίας αισθητήρα

    time = millis(); // Μεταβλητή χρόνο

    Wire.beginTransmission(MPU_addr); // Έναρξη ανάγνωσης τιμών επιταχύνσεων ανά άξονα
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_addr, 6, true);
    int16_t XAxisFull =  (Wire.read() << 8 | Wire.read()); // Ανάγνωση τιμής επιτάχυνσης στον άξονα X
    int16_t YAxisFull =  (Wire.read() << 8 | Wire.read());
    int16_t ZAxisFull =  (Wire.read() << 8 | Wire.read());
    float XAxisFinal = (float) XAxisFull / 16384; // Για τιμές επιτάχυνσης από -2 g μέχρι + 2 g οι τιμές του αισθητήρα διαιρούνται με το 16384 σύμφωνα με τον κατασκευαστή
    float YAxisFinal = (float)YAxisFull / 16384;
    float ZAxisFinal = (float) ZAxisFull / 16384;
```

```
        Serial.print(time); // Εκτύπωση τιμών σειριακά στο Serial Monitor
        Serial.print(" accX=");
        Serial.print(XAxisFinal);
        Serial.print(" accY=");
        Serial.print(YAxisFinal);
        Serial.print(" accZ=");
        Serial.print(ZAxisFinal);
        Serial.println(" ");

        delay(3); // Αναμονή 3 milliseconds μέχρι την επόμενη επανάληψη
}
```

Παράρτημα Β. Κώδικας επικοινωνίας μεταξύ αισθητήρα και Python

```
import matplotlib.pyplot as plt   # Ορισμός βιβλιοθηκών που εμπεριέχονται στον κώδικα
import matplotlib.animation as animation
import serial
import re
import time
import math
from scipy import fftpack
import heapq
import numpy as np
from sklearn import linear_model
from scipy.signal import find_peaks

ser = serial.Serial('com6', 500000)

# Ορισμός των τιμών επιτάχυνσης σε κάθε άξονα στη θέση ισορροπίας του φορέα ως μέση τιμή
mean_acc_x = -0.1608
mean_acc_y = -0.0406
mean_acc_z = 1.0498

# Παράμετροι για τη παρουσίαση των τιμών του αισθητήρα σε πραγματικό χρόνο
x_len = 400         # Number of points to display
y_range = [-2, 2]  # Range of possible Y values to display

# Δεδομένα γραφήματος τιμών σε πραγματικό χρόνο
fig = plt.figure()
ax = fig.add_subplot(2, 2, 1)
ay = fig.add_subplot(2, 2, 2)
az = fig.add_subplot(2, 2, 3)
xs = list(range(0, 400))
ys_x = [mean_acc_x] * x_len
ys_y = [mean_acc_y] * x_len
ys_z = [mean_acc_z] * x_len
ax.set_ylim(y_range)
ax.set_xticklabels([])
ay.set_ylim(y_range)
ay.set_xticklabels([])
az.set_ylim(y_range)
az.set_xticklabels([])

# Δημιουργία λίστας τιμών που θα ανανεώνεται σε κάθε επανάληψη
line_x, = ax.plot(xs, ys_x)
line_y, = ay.plot(xs, ys_y)
```

```
line_z, = az.plot(xs, ys_z)

# Τίτλοι γραφημάτων
ax.set_title('X Acceleration (g)')
ay.set_title('Y Acceleration (g)')
az.set_title('Z Acceleration (g)')

# Συνάρτηση που καλείται από την εντολή FuncAnimation

def animate(i, ys_x, ys_y, ys_z):

    # Ανάγνωση της επιτάχυνσης στον άξονα X σειριακά
    arduinoData = ser.readline().decode('ascii')

    accX = re.search('accX', arduinoData)
    x = int(accX.start()) + 5
    accX = arduinoData[x:x + 8]
    search = re.search(' ', accX)
    end = search.start()
    accX = float(accX[0:end])

    # Προσθήκη της τιμής στην αντίστοιχη λίστα
    ys_x.append(accX)

    # Περιορισμός μεγέθους λίστας
    ys_x = ys_x[-x_len:]

    # Ανανέωση όλων των τιμών της λίστας
    line_x.set_ydata(ys_x)

    # Ομοίως για τις τιμές στους άξονες Y και Z
    accY = re.search('accY', arduinoData)
    x = int(accY.start()) + 5
    accY = arduinoData[x:x + 8]
    search = re.search(' ', accY)
    end = search.start()
    accY = float(accY[0:end])

    ys_y.append(accY)

    ys_y = ys_y[-x_len:]

    line_y.set_ydata(ys_y)

    accZ = re.search('accZ', arduinoData)
    x = int(accZ.start()) + 5
    accZ = arduinoData[x:x + 8]
    search = re.search(' ', accZ)
    end = search.start()
    accZ = float(accZ[0:end])
```

```
    ys_z.append(accZ)

    ys_z = ys_z[-x_len:]

    line_z.set_ydata(ys_z)

    return line_x, line_y, line_z,

# Ορισμός του γραφήματος ώστε να καλεί τη συνάρτηση animate() περιοδικά
start = time.time()
ani = animation.FuncAnimation(fig, animate, fargs=(ys_x, ys_y, ys_z,), interval=0.001,
blit=True)
plt.show()

end = time.time()

ys_x = ys_x[400:len(ys_x)]
ys_y = ys_y[400:len(ys_y)]
ys_z = ys_z[400:len(ys_z)]
print('\nTotal number of samples: ', len(ys_z))
duration = end - start
print('Duration: ', '%0.2f' % duration, 'sec')
fs = len(ys_z) / duration
print('Sampling frequency: ', '%0.0f' % fs, 'Hz')
max_acc_x = mean_acc_x
min_acc_x = mean_acc_x
min_pos_x = -1
max_pos_x = -1
for i in range(0, len(ys_x)):
    if ys_x[i] >= max_acc_x:
        max_acc_x = ys_x[i]
        max_pos_x = i
    if ys_x[i] <= min_acc_x:
        min_acc_x = ys_x[i]
        min_pos_x = i
print('Maximum acceleration in X axis: ', max_acc_x, ' g')
max_acc_y = mean_acc_y
min_acc_y = mean_acc_y
min_pos_y = -1
max_pos_y = -1
for i in range(0, len(ys_y)):
    if ys_y[i] >= max_acc_y:
        max_acc_y = ys_y[i]
        max_pos_y = i
    if ys_y[i] <= min_acc_y:
        min_acc_y = ys_y[i]
        min_pos_y = i
print('Maximum acceleration in Y axis: ', max_acc_y, ' g')
max_acc_z = mean_acc_z
min_acc_z = mean_acc_z
```

```
min_pos_z = -1
max_pos_z = -1
for i in range(0, len(ys_z)):
    if ys_z[i] >= max_acc_z:
        max_acc_z = ys_z[i]
        max_pos_z = i
    if ys_z[i] <= min_acc_z:
        min_acc_z = ys_z[i]
        min_pos_z = i
print('Maximum acceleration in Z axis: ', max_acc_z, ' g')

# Αποθήκευση καταγραφής
data = open("accelerations_data.txt", "w")
f = open("accelerations_data.txt", 'a')
f.write('Time(sec)          AccX(g/s)          AccY(g/s)        AccZ(deg/s)')
time_step = 1 / fs
time_step = str('%0.4f' % time_step)
time_step = float(time_step)
for i in range(0, len(ys_x)):
    f.write('\n')
    time = i * time_step
    f.write(str('%0.4f' % time))
    f.write('              ')
    f.write(str(ys_x[i]))
    f.write('              ')
    f.write(str(ys_y[i]))
    f.write('              ')
    f.write(str(ys_z[i]))

results = open("results.txt", "w")
g = open("results.txt", 'a')
g.write('Results:\n \n')

# Εύρεση κύριων συχνοτήτων στους άξονες X, Y, Z αντίστοιχα

X_ = fftpack.fft(ys_x)   # Εκτέλεση του Γρήγορου Μετασχηματισμού  Fourier
freqs = fftpack.fftfreq(len(ys_x)) * fs

f_res = fs / len(ys_z)
print('\nFFT Results: \n(Frequency resolution: ', '%0.2f' % f_res, ' Hz)')

y_max = heapq.nlargest(2, np.abs(X_))
x_max = max(freqs)
for i in range(0, len(ys_x)):
    a = freqs[i]
    b = np.abs(X_)[i]
    if b == y_max[1] and a > 0:
        maxf_x = a
        print('\n    Main frequency in X axis: ', '%0.2f' % a, ' Hz')
        g.write('Main frequency in X axis: ')
```

```
        g.write('%0.2f' % a)
        g.write(' Hz')
        g.write('\n')

Y_ = fftpack.fft(ys_y)
freqs = fftpack.fftfreq(len(ys_y)) * fs
y_max = heapq.nlargest(2, np.abs(Y_))
x_max = max(freqs)
for i in range(0, len(ys_y)):
    a = freqs[i]
    b = np.abs(Y_)[i]
    if b == y_max[1] and a > 0:
        maxf_y = a
        print('    Main frequency in Y axis: ', '%0.2f' % a, ' Hz')
        g.write('Main frequency in Y axis: ')
        g.write('%0.2f' % a)
        g.write(' Hz')
        g.write('\n')

Z_ = fftpack.fft(ys_z)
freqs = fftpack.fftfreq(len(ys_z)) * fs

y_max = heapq.nlargest(2, np.abs(Z_))
x_max = max(freqs)
for i in range(0, len(ys_z)):
    a = freqs[i]
    b = np.abs(Z_)[i]
    if b == y_max[1] and a > 0:
        maxf_z = a
        print('    Main frequency in Z axis: ', '%0.2f' % a, ' Hz\n')
        g.write('Main frequency in Z axis: ')
        g.write('%0.2f' % a)
        g.write(' Hz')
        g.write('\n \n')

fig2 = plt.figure()
ax = fig2.add_subplot(2, 2, 1)
ay = fig2.add_subplot(2, 2, 2)
az = fig2.add_subplot(2, 2, 3)

ax.set_title('X Acceleration (g)')
ax.set_xticklabels([])
ax.plot(ys_x)

ay.set_title('Y Acceleration (g)')
ay.set_xticklabels([])
ay.plot(ys_y)

az.set_title('Z Acceleration (g)')
az.set_xticklabels([])
```

```
az.plot(ys_z)

plt.show()

# Αφαίρεση των τιμών επιτάχυνσης που δεν αποτελούν μέρος της ταλάντωσης

if max_pos_x != -1 and min_pos_x != -1:
    if max_pos_x < min_pos_x:
        ys_x = ys_x[max_pos_x - 2:len(ys_x)]
    else:
        ys_x = ys_x[min_pos_x - 2:len(ys_x)]
if max_pos_y != -1 and min_pos_y != -1:
    if max_pos_y < min_pos_y:
        ys_y = ys_y[max_pos_y - 2:len(ys_y)]
    else:
        ys_y = ys_y[min_pos_y - 2:len(ys_y)]
if max_pos_z != -1 and min_pos_z != -1:
    if max_pos_z < min_pos_z:
        ys_z = ys_z[max_pos_z - 2:len(ys_z)]
    else:
        ys_z = ys_z[min_pos_z - 2:len(ys_z)]


# Απόσβεση στο άξονα X

if max_pos_x != -1 and min_pos_x != -1:
    ys1 = ys_x[0:len(ys_x) // 4]  # Διαχωρισμός των τιμών σε τέσσερα πακέτα για καλύτερη λειτουργία της βιβλιοθήκης για τηνεύρεση ακροτάτων τιμών σε γράφημα
    ys2 = ys_x[len(ys_x) // 4:len(ys_x) * 2 // 4]
    ys3 = ys_x[len(ys_x) * 2 // 4:len(ys_x) * 3 // 4]
    ys4 = ys_x[len(ys_x) * 3 // 4:len(ys_x)]
    prom1 = 1.4 * (max(ys2) - mean_acc_x)  # Υπολογισμός του καταλληλότερου εύρους για εύρεση ακροτάτων σύμφωνα με την χρησιμοποιούμενη βιβλιοθήκη
    prom2 = 1.4 * (max(ys3) - mean_acc_x)
    prom3 = 1.4 * (max(ys4) - mean_acc_x)
    prom4 = 3 / 4 * prom3
    ys_x = np.array(ys_x)
    ys1 = np.array(ys1)
    ys2 = np.array(ys2)
    ys3 = np.array(ys3)
    ys4 = np.array(ys4)
    ys1_ = -ys1
    ys2_ = -ys2
    ys3_ = -ys3
    ys4_ = -ys4
    peaks1, _ = find_peaks(ys1, prominence=prom1)
    peaks2, _ = find_peaks(ys2, prominence=prom2)
    peaks3, _ = find_peaks(ys3, prominence=prom3)
    peaks4, _ = find_peaks(ys4, prominence=prom4)
    peaks2 = peaks2 + len(ys_x) // 4
```

```
    peaks3 = peaks3 + len(ys_x) * 2 // 4
    peaks4 = peaks4 + len(ys_x) * 3 // 4
    peaks_x = []  # Αποθήκευση ακροτάτων σε μια λίστα τιμών
    for i in peaks1:
        if ys_x[i] > mean_acc_x + 0.02:
            peaks_x.append(i)
    for i in peaks2:
        if ys_x[i] > mean_acc_x + 0.02:
            peaks_x.append(i)
    for i in peaks3:
        if ys_x[i] > mean_acc_x + 0.02:
            peaks_x.append(i)
    for i in peaks4:
        if ys_x[i] > mean_acc_x + 0.02:
            peaks_x.append(i)

    if min_pos_x < max_pos_x and len(peaks_x) != 0:
        for i in range(0, len(peaks_x)):
            if ys_x[peaks_x[i]] == max_acc_x:
                max_p = i

        peaks_x = peaks_x[max_p:len(peaks_x)]  # Ορισμός του πρώτου ακρότατου ως αυτό με τη μεγαλύτερη απόλυτη τιμή

    peaks1_, _ = find_peaks(ys1_, prominence=prom1)
    peaks2_, _ = find_peaks(ys2_, prominence=prom2)
    peaks3_, _ = find_peaks(ys3_, prominence=prom3)
    peaks4_, _ = find_peaks(ys4_, prominence=prom4)
    peaks2_ = peaks2_ + len(ys_x) // 4
    peaks3_ = peaks3_ + len(ys_x) * 2 // 4
    peaks4_ = peaks4_ + len(ys_x) * 3 // 4
    peaks_x_ = []
    for i in peaks1_:
        if ys_x[i] < mean_acc_x - 0.02:
            peaks_x_.append(i)
    for i in peaks2_:
        if ys_x[i] < mean_acc_x - 0.02:
            peaks_x_.append(i)
    for i in peaks3_:
        if ys_x[i] < mean_acc_x - 0.02:
            peaks_x_.append(i)
    for i in peaks4_:
        if ys_x[i] < mean_acc_x - 0.02:
            peaks_x_.append(i)

    if min_pos_x > max_pos_x and len(peaks_x_) != 0:
        for i in range(0, len(peaks_x_)):
            if ys_x[peaks_x_[i]] == min_acc_x:
                min_p = i
```

```
        peaks_x_ = peaks_x_[min_p:len(peaks_x_)]


# Απόσβεση στο άξονα Y ομοίως με τον X

if max_pos_y != -1 and min_pos_y != -1:
    ys1 = ys_y[0:len(ys_y) // 4]
    ys2 = ys_y[len(ys_y) // 4:len(ys_y) * 2 // 4]
    ys3 = ys_y[len(ys_y) * 2 // 4:len(ys_y) * 3 // 4]
    ys4 = ys_y[len(ys_y) * 3 // 4:len(ys_y)]
    prom1 = 1.4 * (max(ys2) - mean_acc_y)
    prom2 = 1.4 * (max(ys3) - mean_acc_y)
    prom3 = 1.4 * (max(ys4) - mean_acc_y)
    prom4 = 3 / 4 * prom3
    ys_y = np.array(ys_y)
    ys1 = np.array(ys1)
    ys2 = np.array(ys2)
    ys3 = np.array(ys3)
    ys4 = np.array(ys4)
    ys1_ = -ys1
    ys2_ = -ys2
    ys3_ = -ys3
    ys4_ = -ys4

    peaks1, _ = find_peaks(ys1, prominence=prom1)
    peaks2, _ = find_peaks(ys2, prominence=prom2)
    peaks3, _ = find_peaks(ys3, prominence=prom3)
    peaks4, _ = find_peaks(ys4, prominence=prom4)
    peaks2 = peaks2 + len(ys_y) // 4
    peaks3 = peaks3 + len(ys_y) * 2 // 4
    peaks4 = peaks4 + len(ys_y) * 3 // 4
    peaks_y = []
    for i in peaks1:
        if ys_y[i] > mean_acc_y + 0.02:
            peaks_y.append(i)
    for i in peaks2:
        if ys_y[i] > mean_acc_y + 0.02:
            peaks_y.append(i)
    for i in peaks3:
        if ys_y[i] > mean_acc_y + 0.02:
            peaks_y.append(i)
    for i in peaks4:
        if ys_y[i] > mean_acc_y + 0.02:
            peaks_y.append(i)

    if min_pos_y < max_pos_y and len(peaks_y) != 0:
        for i in range(0, len(peaks_y)):
            if ys_y[peaks_y[i]] == max_acc_y:
                max_p = i
```

```
        peaks_y = peaks_y[max_p:len(peaks_y)]

    peaks1_, _ = find_peaks(ys1_, prominence=prom1)
    peaks2_, _ = find_peaks(ys2_, prominence=prom2)
    peaks3_, _ = find_peaks(ys3_, prominence=prom3)
    peaks4_, _ = find_peaks(ys4_, prominence=prom4)
    peaks2_ = peaks2_ + len(ys_y) // 4
    peaks3_ = peaks3_ + len(ys_y) * 2 // 4
    peaks4_ = peaks4_ + len(ys_y) * 3 // 4
    peaks_y_ = []
    for i in peaks1_:
        if ys_y[i] < mean_acc_y - 0.02:
            peaks_y_.append(i)
    for i in peaks2_:
        if ys_y[i] < mean_acc_y - 0.02:
            peaks_y_.append(i)
    for i in peaks3_:
        if ys_y[i] < mean_acc_y - 0.02:
            peaks_y_.append(i)
    for i in peaks4_:
        if ys_y[i] < mean_acc_y - 0.02:
            peaks_y_.append(i)

    if max_pos_y < min_pos_y and len(peaks_y_) != 0:
        for i in range(0, len(peaks_y_)):
            if ys_y[peaks_y_[i]] == min_acc_y:
                min_p = i

        peaks_y_ = peaks_y_[min_p:len(peaks_y_)]


# Απόσβεση στο άξονα Z ομοίως με τον X

if max_pos_z != -1 and min_pos_z != -1:
    ys1 = ys_z[0:len(ys_z) // 4]
    ys2 = ys_z[len(ys_z) // 4:len(ys_z) * 2 // 4]
    ys3 = ys_z[len(ys_z) * 2 // 4:len(ys_z) * 3 // 4]
    ys4 = ys_z[len(ys_z) * 3 // 4:len(ys_z)]
    prom1 = 1.4 * (max(ys2) - mean_acc_z)
    prom2 = 1.4 * (max(ys3) - mean_acc_z)
    prom3 = 1.4 * (max(ys4) - mean_acc_z)
    prom4 = 3 / 4 * prom3
    ys_z = np.array(ys_z)
    ys1 = np.array(ys1)
    ys2 = np.array(ys2)
    ys3 = np.array(ys3)
    ys4 = np.array(ys4)
    ys1_ = -ys1
    ys2_ = -ys2
    ys3_ = -ys3
```

```
ys4_ = -ys4

peaks1, _ = find_peaks(ys1, prominence=prom1)
peaks2, _ = find_peaks(ys2, prominence=prom2)
peaks3, _ = find_peaks(ys3, prominence=prom3)
peaks4, _ = find_peaks(ys4, prominence=prom4)
peaks2 = peaks2 + len(ys_z) // 4
peaks3 = peaks3 + len(ys_z) * 2 // 4
peaks4 = peaks4 + len(ys_z) * 3 // 4
peaks = []
for i in peaks1:
    if ys_z[i] > mean_acc_z + 0.02:
        peaks.append(i)
for i in peaks2:
    if ys_z[i] > mean_acc_z + 0.02:
        peaks.append(i)
for i in peaks3:
    if ys_z[i] > mean_acc_z + 0.02:
        peaks.append(i)
for i in peaks4:
    if ys_z[i] > mean_acc_z + 0.02:
        peaks.append(i)

if min_pos_z < max_pos_z and len(peaks) != 0:
    for i in range(0, len(peaks)):
        if ys_z[peaks[i]] == max_acc_z:
            max_p = i

    peaks = peaks[max_p:len(peaks)]

peaks1_, _ = find_peaks(ys1_, prominence=prom1)
peaks2_, _ = find_peaks(ys2_, prominence=prom2)
peaks3_, _ = find_peaks(ys3_, prominence=prom3)
peaks4_, _ = find_peaks(ys4_, prominence=prom4)
peaks2_ = peaks2_ + len(ys_z) // 4
peaks3_ = peaks3_ + len(ys_z) * 2 // 4
peaks4_ = peaks4_ + len(ys_z) * 3 // 4
peaks_ = []
for i in peaks1_:
    if ys_z[i] < mean_acc_z - 0.02:
        peaks_.append(i)
for i in peaks2_:
    if ys_z[i] < mean_acc_z - 0.02:
        peaks_.append(i)
for i in peaks3_:
    if ys_z[i] < mean_acc_z - 0.02:
        peaks_.append(i)
for i in peaks4_:
    if ys_z[i] < mean_acc_z - 0.02:
        peaks_.append(i)
```

```
    if max_pos_z < min_pos_z and len(peaks_) != 0:
        for i in range(0, len(peaks_)):
            if ys_z[peaks_[i]] == min_acc_z:
                min_p = i

        peaks_ = peaks_[min_p:len(peaks_)]

# Ορισμός από τον χρήστη τους άξονες για τους οποίους θα υπολογιστούν οι συντελεστές απόσβεσης

key1 = 0
if max_pos_x != -1 and min_pos_x != -1:
    key1 = input('Type 1 to calculate damping in X axis:')
key2 = 0
if max_pos_y != -1 and min_pos_y != -1:
    key2 = input('Type 1 to calculate damping in Y axis:')
key3 = 0
if max_pos_z != -1 and min_pos_z != -1:
    key3 = input('Type 1 to calculate damping in Z axis:')

if key1 == '1':
    damp_x = ys_x[peaks_x]
    for i in range(0, len(damp_x)):
        damp_x[i] = - math.log((damp_x[i] - mean_acc_x) / (max_acc_x - mean_acc_x))
    damp_x = np.reshape(damp_x, (-1, 1))
    f = peaks_x[0]
    for i in range(0, len(peaks_x)):
        peaks_x[i] = (peaks_x[i] - f) * 2 * math.pi * maxf_x / fs
    peaks_x = np.reshape(peaks_x, (-1, 1))

    regr = linear_model.LinearRegression()
    regr.fit(peaks_x, damp_x)
    pred_x = regr.predict(peaks_x)
    ks_x = float((pred_x[-1] - pred_x[0]) / (peaks_x[-1] - peaks_x[0]) * 100)
    b_x = pred_x[0] - ks_x * peaks_x[0]

    dist_x = []
    for i in range(0, len(peaks_x)):
        dx = (peaks_x[i] / (ks_x / 100) + damp_x[i] - b_x) / ((ks_x / 100) + 1 / (ks_x / 100))
        dy = (ks_x / 100) * dx + b_x
        d = math.sqrt((dx - peaks_x[i]) ** 2 + (dy - damp_x[i]) ** 2)
        dist_x.append(d)

    mean_d = np.mean(dist_x)
    s = math.sqrt(np.var(dist_x))
    low = mean_d - 2 * s
    high = mean_d + 2 * s

    n_peaks_x = []
```

```
    n_damp_x = []
    for i in range(0, len(dist_x)):
        if low <= dist_x[i] <= high:
            n_peaks_x.append(peaks_x[i])
            n_damp_x.append(damp_x[i])

    regr = linear_model.LinearRegression()
    regr.fit(n_peaks_x, n_damp_x)
    n_pred_x = regr.predict(n_peaks_x)
    n_ks_x = float((n_pred_x[-1] - n_pred_x[0]) / (n_peaks_x[-1] - n_peaks_x[0]) * 100)
    print('\n1st Damping value in X axis: ', '%0.2f' % n_ks_x, ' %')
    g.write('1st Damping value in X axis: ')
    g.write('%0.2f' % n_ks_x)
    g.write(' % \n')

    damp_x_ = ys_x[peaks_x_]
    for i in range(0, len(damp_x_)):
        damp_x_[i] = - math.log(abs((damp_x_[i] - mean_acc_x) / (min_acc_x -
mean_acc_x)))
    damp_x_ = np.reshape(damp_x_, (-1, 1))
    f = peaks_x_[0]
    for i in range(0, len(peaks_x_)):
        peaks_x_[i] = (peaks_x_[i] - f) * 2 * math.pi * maxf_x / fs
    peaks_x_ = np.reshape(peaks_x_, (-1, 1))

    regr = linear_model.LinearRegression()
    regr.fit(peaks_x_, damp_x_)
    pred_x_ = regr.predict(peaks_x_)
    ks_x_ = float((pred_x_[-1] - pred_x_[0]) / (peaks_x_[-1] - peaks_x_[0]) * 100)
    b_x_ = pred_x_[0] - ks_x_ * peaks_x_[0]

    dist_x_ = []
    for i in range(0, len(peaks_x_)):
        dx = (peaks_x_[i] / (ks_x_ / 100) + damp_x_[i] - b_x_) / ((ks_x_ / 100) + 1 / (ks_x_ /
100))
        dy = (ks_x_ / 100) * dx + b_x_
        d = math.sqrt((dx - peaks_x_[i]) ** 2 + (dy - damp_x_[i]) ** 2)
        dist_x_.append(d)

    mean_d = np.mean(dist_x_)
    s = math.sqrt(np.var(dist_x_))
    low = mean_d - 2 * s
    high = mean_d + 2 * s

    n_peaks_x_ = []
    n_damp_x_ = []
    for i in range(0, len(dist_x_)):
        if low <= dist_x_[i] <= high:
            n_peaks_x_.append(peaks_x_[i])
            n_damp_x_.append(damp_x_[i])
```

```
    regr = linear_model.LinearRegression()
    regr.fit(n_peaks_x_, n_damp_x_)
    n_pred_x_ = regr.predict(n_peaks_x_)
    n_ks_x_ = float((n_pred_x_[-1] - n_pred_x_[0]) / (n_peaks_x_[-1] - n_peaks_x_[0]) *
100)
    print('2nd Damping value in X axis: ', '%0.2f' % n_ks_x_, ' %')
    g.write('2nd Damping value in X axis: ')
    g.write('%0.2f' % n_ks_x_)
    g.write(' % \n')

    mean_ks_x = (n_ks_x + n_ks_x_) / 2
    print('Mean damping value in X axis:', '%0.2f' % mean_ks_x, ' %')
    g.write('Mean Damping value in X axis: ')
    g.write('%0.2f' % mean_ks_x)
    g.write(' % \n \n')

    fig1 = plt.figure()
    plt1 = fig1.add_subplot(1, 1, 1)
    plt1.set_title('Damping calculation in X axis')
    plt1.scatter(n_peaks_x, n_damp_x)
    plt1.scatter(n_peaks_x_, n_damp_x_)
    plt1.plot(n_peaks_x, n_pred_x, label='1st Value')
    plt1.plot(n_peaks_x_, n_pred_x_, label='2nd Value')
    plt1.legend()


if key2 == '1':
    damp_y = ys_y[peaks_y]
    for i in range(0, len(damp_y)):
        damp_y[i] = - math.log((damp_y[i] - mean_acc_y) / (max_acc_y - mean_acc_y))
    damp_y = np.reshape(damp_y, (-1, 1))
    f = peaks_y[0]
    for i in range(0, len(peaks_y)):
        peaks_y[i] = (peaks_y[i] - f) * 2 * math.pi * maxf_y / fs
    peaks_y = np.reshape(peaks_y, (-1, 1))

    regr = linear_model.LinearRegression()
    regr.fit(peaks_y, damp_y)
    pred_y = regr.predict(peaks_y)
    ks_y = float((pred_y[-1] - pred_y[0]) / (peaks_y[-1] - peaks_y[0]) * 100)
    b_y = pred_y[0] - ks_y * peaks_y[0]

    dist_y = []
    for i in range(0, len(peaks_y)):
        dx = (peaks_y[i] / (ks_y / 100) + damp_y[i] - b_y) / ((ks_y / 100) + 1 / (ks_y / 100))
        dy = (ks_y / 100) * dx + b_y
        d = math.sqrt((dx - peaks_y[i]) ** 2 + (dy - damp_y[i]) ** 2)
        dist_y.append(d)
```

```
    mean_d = np.mean(dist_y)
    s = math.sqrt(np.var(dist_y))
    low = mean_d - 2 * s
    high = mean_d + 2 * s

    n_peaks_y = []
    n_damp_y = []
    for i in range(0, len(dist_y)):
        if low <= dist_y[i] <= high:
            n_peaks_y.append(peaks_y[i])
            n_damp_y.append(damp_y[i])

    regr = linear_model.LinearRegression()
    regr.fit(n_peaks_y, n_damp_y)
    n_pred_y = regr.predict(n_peaks_y)
    n_ks_y = float((n_pred_y[-1] - n_pred_y[0]) / (n_peaks_y[-1] - n_peaks_y[0]) * 100)
    print('\n1st Damping value in Y axis: ', '%0.2f' % n_ks_y, ' %')
    g.write('1st Damping value in Y axis: ')
    g.write('%0.2f' % n_ks_y)
    g.write(' % \n')

    damp_y_ = ys_y[peaks_y_]
    for i in range(0, len(damp_y_)):
        damp_y_[i] = - math.log(abs((damp_y_[i] - mean_acc_y) / (min_acc_y -
mean_acc_y)))
    damp_y_ = np.reshape(damp_y_, (-1, 1))
    f = peaks_y_[0]
    for i in range(0, len(peaks_y_)):
        peaks_y_[i] = (peaks_y_[i] - f) * 2 * math.pi * maxf_y / fs
    peaks_y_ = np.reshape(peaks_y_, (-1, 1))

    regr = linear_model.LinearRegression()
    regr.fit(peaks_y_, damp_y_)
    pred_y_ = regr.predict(peaks_y_)
    ks_y_ = float((pred_y_[-1] - pred_y_[0]) / (peaks_y_[-1] - peaks_y_[0]) * 100)
    b_y_ = pred_y_[0] - ks_y_ * peaks_y_[0]

    dist_y_ = []
    for i in range(0, len(peaks_y_)):
        dx = (peaks_y_[i] / (ks_y_ / 100) + damp_y_[i] - b_y_) / ((ks_y_ / 100) + 1 / (ks_y_ /
100))
        dy = (ks_y_ / 100) * dx + b_y_
        d = math.sqrt((dx - peaks_y_[i]) ** 2 + (dy - damp_y_[i]) ** 2)
        dist_y_.append(d)

    mean_d = np.mean(dist_y_)
    s = math.sqrt(np.var(dist_y_))
    low = mean_d - 2 * s
    high = mean_d + 2 * s
```

```
    n_peaks_y_ = []
    n_damp_y_ = []
    for i in range(0, len(dist_y_)):
        if low <= dist_y_[i] <= high:
            n_peaks_y_.append(peaks_y_[i])
            n_damp_y_.append(damp_y_[i])

    regr = linear_model.LinearRegression()
    regr.fit(n_peaks_y_, n_damp_y_)
    n_pred_y_ = regr.predict(n_peaks_y_)
    n_ks_y_ = float((n_pred_y_[-1] - n_pred_y_[0]) / (n_peaks_y_[-1] - n_peaks_y_[0]) *
100)
    print('2nd Damping value in Y axis: ', '%0.2f' % n_ks_y_, ' %')
    g.write('2nd Damping value in Y axis: ')
    g.write('%0.2f' % n_ks_y_)
    g.write(' % \n')

    mean_ks_y = (n_ks_y + n_ks_y_) / 2
    print('Mean damping value in Y axis:', '%0.2f' % mean_ks_y, ' %')
    g.write('Mean Damping value in Y axis: ')
    g.write('%0.2f' % mean_ks_y)
    g.write(' % \n \n')

    fig2 = plt.figure()
    plt2 = fig2.add_subplot(1, 1, 1)
    plt2.set_title('Damping calculation in Y axis')
    plt2.scatter(n_peaks_y, n_damp_y)
    plt2.scatter(n_peaks_y_, n_damp_y_)
    plt2.plot(n_peaks_y, n_pred_y, label='1st Value')
    plt2.plot(n_peaks_y_, n_pred_y_, label='2nd Value')
    plt2.legend()

if key3 == '1':
    damp = ys_z[peaks]
    for i in range(0, len(damp)):
        damp[i] = - math.log((damp[i] - mean_acc_z) / (max_acc_z - mean_acc_z))
    damp = np.reshape(damp, (-1, 1))
    f = peaks[0]
    for i in range(0, len(peaks)):
        peaks[i] = (peaks[i] - f) * 2 * math.pi * maxf_z / fs
    peaks = np.reshape(peaks, (-1, 1))

    regr = linear_model.LinearRegression()
    regr.fit(peaks, damp)
    pred = regr.predict(peaks)
    ks_z = float((pred[-1] - pred[0]) / (peaks[-1] - peaks[0]) * 100)
    b_z = pred[0] - ks_z * peaks[0]

    dist_z = []
    for i in range(0, len(peaks)):
```

```
    dx = (peaks[i] / (ks_z / 100) + damp[i] - b_z) / ((ks_z / 100) + 1 / (ks_z / 100))
    dy = (ks_z / 100) * dx + b_z
    d = math.sqrt((dx - peaks[i]) ** 2 + (dy - damp[i]) ** 2)
    dist_z.append(d)

mean_d = np.mean(dist_z)
s = math.sqrt(np.var(dist_z))
low = mean_d - 2 * s
high = mean_d + 2 * s
# print(mean_d, s, low, high)

n_peaks = []
n_damp = []
for i in range(0, len(dist_z)):
    if low <= dist_z[i] <= high:
        n_peaks.append(peaks[i])
        n_damp.append(damp[i])

regr = linear_model.LinearRegression()
regr.fit(n_peaks, n_damp)
n_pred = regr.predict(n_peaks)
n_ks_z = float((n_pred[-1] - n_pred[0]) / (n_peaks[-1] - n_peaks[0]) * 100)
print('\n1st Damping value in Z axis: ', '%0.2f' % n_ks_z, ' %')
g.write('1st Damping value in Z axis: ')
g.write('%0.2f' % n_ks_z)
g.write(' % \n')

damp_ = ys_z[peaks_]
for i in range(0, len(damp_)):
    damp_[i] = - math.log(abs((damp_[i] - mean_acc_z) / (min_acc_z - mean_acc_z)))
damp_ = np.reshape(damp_, (-1, 1))
f = peaks_[0]
for i in range(0, len(peaks_)):
    peaks_[i] = (peaks_[i] - f) * 2 * math.pi * maxf_z / fs
peaks_ = np.reshape(peaks_, (-1, 1))

regr = linear_model.LinearRegression()
regr.fit(peaks_, damp_)
pred_ = regr.predict(peaks_)
ks_z_ = float((pred_[-1] - pred_[0]) / (peaks_[-1] - peaks_[0]) * 100)
b_z_ = pred_[0] - ks_z_ * peaks_[0]

dist_z_ = []
for i in range(0, len(peaks_)):
    dx = (peaks_[i] / (ks_z_ / 100) + damp_[i] - b_z_) / ((ks_z_ / 100) + 1 / (ks_z_ / 100))
    dy = (ks_z_ / 100) * dx + b_z_
    d = math.sqrt((dx - peaks_[i]) ** 2 + (dy - damp_[i]) ** 2)
    dist_z_.append(d)

mean_d = np.mean(dist_z_)
```

```
    s = math.sqrt(np.var(dist_z_))
    low = mean_d - 2 * s
    high = mean_d + 2 * s

    n_peaks_ = []
    n_damp_ = []
    for i in range(0, len(dist_z_)):
        if low <= dist_z_[i] <= high:
            n_peaks_.append(peaks_[i])
            n_damp_.append(damp_[i])

    regr = linear_model.LinearRegression()
    regr.fit(n_peaks_, n_damp_)
    n_pred_ = regr.predict(n_peaks_)
    n_ks_z_ = float((n_pred_[-1] - n_pred_[0]) / (n_peaks_[-1] - n_peaks_[0]) * 100)
    print('2nd Damping value in Z axis: ', '%0.2f' % n_ks_z_, ' %')
    g.write('2nd Damping value in Z axis: ')
    g.write('%0.2f' % n_ks_z_)
    g.write(' % \n')

    mean_ks_z = (n_ks_z + n_ks_z_) / 2
    print('Mean damping value in Z axis:', '%0.2f' % mean_ks_z, ' %')
    g.write('Mean Damping value in Z axis: ')
    g.write('%0.2f' % mean_ks_z)
    g.write(' % \n \n')

    fig3 = plt.figure()
    plt3 = fig3.add_subplot(1, 1, 1)
    plt3.set_title('Damping calculation in Z axis')
    plt3.scatter(n_peaks, n_damp)
    plt3.scatter(n_peaks_, n_damp_)
    plt3.plot(n_peaks, n_pred, label='1st Value')
    plt3.plot(n_peaks_, n_pred_, label='2nd Value')
    plt3.legend()

plt.show()
```